NAWCWPNS TP 8340

Using Diagonally Implicit Multistage Integration Methods for Solving Ordinary Differential Equations. Part 1: Introduction and Explicit Methods

by
Jack VanWieren
Research and Technology Group

JANUARY 1997

NAVAL AIR WARFARE CENTER WEAPONS DIVISION
CHINA LAKE, CA 93555-6100

Naval Air Warfare Center Weapons Division

FOREWORD

Diagonally implicit multistage integration methods (DIMSIMs) hold great promise for providing more efficient, more accurate and more robust software for solving systems of ordinary differential equations numerically. The author was supported by Navy In-House Independent Research Funds.

This report was reviewed for technical accuracy by Professor Zdzislaw Jackiewicz, Arizona State University, Tempe, Arizona.

Approved by
R. L. DERR, *Head*
Research and Technology Group
31 January 1997

Under authority of
J. V. CHENEVEY
RADM, U.S. Navy
Commander

Released for publication by
S. HAALAND
Director for Research and Engineering

NAWCWPNS Technical Publication 8340

Published by . . . . Technical Information Division
Collation . . . . Cover, 68 leaves
First printing . . . . 30 copies

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January 1997	Interim report—October 1994-September 1996

4. TITLE AND SUBTITLE

Using Diagonally Implicit Multistage Integration Methods for Solving Ordinary Differential Equations. Part 1: Introduction and Explicit Methods

5. FUNDING NUMBERS

N0001495WX30085
N0001495WX20167

6. AUTHOR(S)

Jack M. Van Wieren

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Air Warfare Center Weapons Division
China Lake, CA 93555-6100

8. PERFORMING ORGANIZATION REPORT NUMBER

NAWCWPNS TP 8340

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Dr. Ronald Derr
Code 4B0000D
Naval Air Warfare Center Weapons Division
China Lake, CA 93555-6100

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12A. DISTRIBUTION/AVAILABILITY STATEMENT

A Statement; public release; distribution unlimited.

12B. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

(U) A survey of theoretical results for the recently invented and highly promising class of diagonally implicit multistage integration methods (DIMSIMs) is provided. New results are obtained that provide improved implementation and simplified analysis. Particular attention is given to issues affecting use on parallel computers in waveform relaxation. The process of implementation parameter derivation and testing for the second and fifth order members of the DIMEX family of ordinary differential equation solver computer codes using explicit DIMSIMs is described . Extensive test results using the standard suite DETEST are reported that show considerable promise. A listing of the FORTRAN 77 research code DIMEX5 is provided.

14. SUBJECT TERMS

Differential Equations, DIMSIMs, Waveform Relaxation, Parallel Computing, Numerical Analysis, Mathematical Software

15. NUMBER OF PAGES 136

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18
298-102

Standard Form 298 Back (Rev. 2-89)

CONTENTS

Introduction 3

Implementing DIMSIMs 19

Techniques for Obtaining Starting Values For DIMSIMs 42

Developing Explicit DIMSIM ODE Solvers 55

The DIMEX Family of Explicit DIMSIM ODE Solvers 89

References 113

Appendix
DIMEX5 Code 117

INTRODUCTION TO DIMSIMS

LITERATURE SURVEY

Diagonally implicit multistage integration methods (DIMSIMs) were first described in a 1992 paper by John Butcher (Reference 1) in which he laid out the essential elements of a new family of general linear methods. His stated purpose was to overcome the glaring weaknesses of existing methods, that is, lack of A-stability for high order linear multistep methods and low stage order and high implementation costs for A-stable implicit Runge-Kutta methods. These methods are diagonally implicit and hence have computational complexity properties similar to diagonally implicit Runge-Kutta (DIRK) methods, but utilize additional parameters generated through the general linear design to overcome the stage order (and hence stiff order) limitations of DIRK methods. Explicit DIMSIMs are not technically "diagonally implicit", since the diagonal is actually 0, but have an advantage over Runge-Kutta methods in that the order barriers for explicit Runge-Kutta methods do not apply, and p-stage methods of order p do indeed exist for all positive integers p. The original concept called for stage order q to equal the number of internal stages s, the number of external stages r, and the order p. In a subsequent paper with Jackiewicz (Reference 2) the family of DIMSIMs was extended to include adjacent methods for which $s + 1 = r = q$, $p = q$ or $q + 1$, $s = r + 1 = q$, $p = q$ or $q + 1$, and $s = r = q$, $p = q + 1$. A significant step toward practical utilization was taken with another paper by Butcher and Jackiewicz (Reference 3) that lays out techniques for error estimation, interpolation, and step-size changing. Jackiewicz, Vermiglio, and Zennaro (Reference 4) devised an alternative step-size changing strategy and showed how incorporation of an additional external stage could provide a satisfactory continuous method. In a separate paper (Reference 5) they also showed that there exist explicit DIMSIMs with regularity properties not possessed by explicit Runge-Kutta methods. Butcher, Chartier, and Jackiewicz, in an unpublished manuscript, "Nordsieck representation of DIMSIMs," recently proposed an alternative representation of DIMSIMs with promise of simplifying analysis and implementation. Both explicit and implicit DIMSIMs up to the order 8 have now been found with appropriate stability properties and were announced by Butcher, Jackiewicz, and Mittelmann (Reference 6), extending techniques described in earlier papers by Butcher and Jackiewicz (Reference 7 and 8).

BASIC DEFINITIONS AND RELATIONSHIPS

We consider the initial value problem:

$$\frac{dy}{dt} = f(t,y), \; y(t_0) = y_0, \; t \in [t_0, T]. \tag{1}$$

We define matrices A, U, B and V such that A is s x s, V is r x r, U is s x r, and B is r x s. Let Y be composed of the s internal stages, F be composed of the s stage derivatives ($F_j = f(t_n + c_j h, Y_j)$) and $y^{[n]}$ be composed of the r external stage values. Then if h is the step size, the solution advances one step through the relationships:

$$
\begin{aligned}
Y_i &= h \sum_{j=1}^{s} a_{ij} F_j + \sum_{j=1}^{r} u_{ij} y_j^{[n-1]}, \quad i = 1,2,\ldots,s \\
y_i^{[n]} &= h \sum_{j=1}^{s} b_{ij} F_j + \sum_{j=1}^{r} v_{ij} y_j^{[n-1]}, \quad i = 1,2,\ldots,r \\
n &= 1,2,\ldots,N, \quad Nh = T - t_0.
\end{aligned} \tag{2}
$$

The external stages are defined through Taylor expansion so that if

$$
y_i^{[n-1]} = \sum_{j=0}^{p} \alpha_{ij} h^j y^{(j)}(t_{n-1}) + O\left(h^{p+1}\right), \tag{3}
$$

then we must have, for some constants α_{ij},

$$
y_i^{[n]} = \sum_{j=0}^{p} \alpha_{ij} h^j y^{(j)}(t_n) + O\left(h^{p+1}\right), \tag{4}
$$

for a method of order p. Butcher has observed (Reference 9) that the effect then is to calculate r neighboring trajectories and to use these to determine solution values. The s values c_i are chosen initially and other method parameters are then determined in a way to produce high stage order, that is, so that:

$$
Y_i = y(t_{n-1} + hc_i) + O\left(h^{q+1}\right), \quad i = 1,2,\ldots,s. \tag{5}
$$

where q is defined to be the stage order.

The description "diagonally implicit" comes from the form of the matrix A, and the following discussion is true for both DIMSIMs and DIRK methods. If we restrict A to be

lower triangular with a constant along the diagonal, the complexity of the solution of the system of nonlinear equations determining Y in Equation 2 reduces greatly. If A is dense and a standard gaussian elimination approach is used in a modified Newton method, the arithmetic complexity is $O((Ms)^3)$, or $O(M^3s^3)$, where s is the number of stages of the method and M is the dimension of the initial value problem. Simply requiring A to be lower triangular separates the system of Ms simultaneous nonlinear equations into s systems of M simultaneous equations to be solved in sequence, each with arithmetic complexity $O(M^3)$ for total complexity $O(sM^3)$, a reduction by a factor of $O(s^2)$.

The advantage of having a single value along the diagonal may be seen from a closer examination of the nonlinear system solution process, which typically involves a modified Newton method. The Newton iteration to solve the nonlinear system $g(x) = 0$ takes the form

$$x_{n+1} = x_n - J(x_n)^{-1} g(x_n), \tag{6}$$

where J is the Jacobian of g. Of course the inverse of the Jacobian is not actually calculated and instead a technique such as gaussian elimination is utilized, and the process followed is to calculate a correction:

$$\begin{aligned} J(x_n)\delta_{n+1} &= -g(x_n), \\ x_{n+1} &= x_n + \delta_{n+1}. \end{aligned} \tag{7}$$

An LU or PLU factorization of $J(x_n)$ is called for here at each iteration, which is $O(M^3)$, where M is the size of the system, and this is the most time consuming step. In practice a new Jacobian is evaluated and a new matrix factorization is carried out only when convergence seems too slow. In the case of DIMSIMs the equation for Y takes the form

$$Y_j = h\sum_{k=1}^{j-1} a_{jk} f(t_{n-1} + c_k h, Y_k) + h a_{jj} f(t_{n-1} + c_j h, Y_j) + y_j^{[n-1]}, \tag{8}$$

with U=I as is usually the case. Then we are solving an equation of the form $g_j(Y_j)=0$, where g_j takes the form:

$$g_j(Y_j) = Y_j - h a_{jj} f(t_{n-1} + c_j h, Y_j) - h\sum_{k=1}^{j-1} a_{jk} f(t_{n-1} + c_k h, Y_k) - y_j^{[n-1]}. \tag{9}$$

All the dependency on Y_j is contained in the first two terms and so only these terms affect the calculation of the Jacobian. Now if all diagonal elements a_{jj} are the same the Jacobians J_j will vary from stage to stage only with the solution, and new Jacobians and their LU decompositions will only have to be computed when convergence seems too slow, which should be relatively rare for small step size h. This can result in substantial savings.

A similar reduction follows from utilization of a nonsingular A (perhaps dense) with a single eigenvalue, and this has been applied in the implicit Runge-Kutta code STRIDE developed by Burrage, Butcher, and Chipman (Reference 10). Because of the work required for linear transformation, this is to be avoided if possible due to the number of transformations that become necessary. Of course the low stage order of DIRK methods, which drastically reduces the observed order of the method for stiff equations to second order, made this alternative approach attractive for use in STRIDE.

The conditions (Equations 2 through 5) may be re-expressed in a convenient form as follows. Let W be the r x (p + 1) matrix of α_{ij} values, and we denote the vector consisting of the kth column of W as α_{k-1}.Let Z be a vector with element $Z_j = z^{j-1}$. Then define w(z) = WZ. Furthermore we may define e^{cz} as the vector

$$e^{cz} = \begin{bmatrix} e^{c_1 z} \\ e^{c_2 z} \\ \vdots \\ e^{c_s z} \end{bmatrix}. \tag{10}$$

Then the following theorem may be used in determining the coefficients of the method:

Theorem 1: A DIMSIM (Equation 2) has order p and stage order p if and only if

$$\begin{aligned} e^{cz} &= zAe^{cz} + Uw(z) + O\left(z^{p+1}\right), \\ e^{z}w(z) &= zBe^{cz} + Vw(z) + O\left(z^{p+1}\right). \end{aligned} \tag{11}$$

Proof: The proof is given by Butcher (Reference 1) and is included here for the sake of completeness.

The stage order condition gives

$$Y_i = y(t_{n-1} + hc_i) + O\left(h^{p+1}\right).$$

We apply the derivative function f to both sides and find that

$$\begin{aligned} f(t_{n-1} + c_i h, Y_i) &= f\left(t_{n-1} + c_i h, y(t_{n-1} + c_i h) + O\left(h^{p+1}\right)\right) \\ &= f\left(t_{n-1} + c_i h, y(t_{n-1} + c_i h)\right) + O\left(h^{p+1}\right) \\ &= y'\left(t_{n-1} + c_i h\right) + O\left(h^{p+1}\right). \end{aligned}$$

We then may write, using Taylor expansion,

$$hf(t_{n-1} + c_i h, Y_i) = \sum_{j=1}^{p} \frac{c_i^{j-1}}{(j-1)!} y^{(j)}(t_{n-1}) h^j + O\left(h^{p+1}\right).$$

Since

$$y_i^{[n]} = \sum_{j=0}^{p} \alpha_{ij} y^{(j)}(t_n) h^j + O\left(h^{p+1}\right)$$

we find through Taylor expansion that

$$y_i^{[n]} = \sum_{j=0}^{p} \left(\sum_{k=0}^{j} \frac{1}{k!} \alpha_{i,j-1} \right) y^{(j)}(t_{n-1}) h^j + O\left(h^{p+1}\right).$$

We now utilize Equation 2, provide a Taylor expansion for each Y_j, and use W to express each $y_j^{[n-1]}$ in terms of $h^k y^{(k)}(t_{n-1})$. Then we can combine coefficients to obtain the two equations

$$\sum_{k=0}^{p}\left(c_i^k - \sum_{j=1}^{s} k a_{ij} c_j^{k-1} - \sum_{j=1}^{r} u_{ij}\alpha_{ij} k!\right)\frac{h^k}{k!} y^{(k)}(t_{n-1}) = O\left(h^{p+1}\right)$$

$$\sum_{k=0}^{p}\left(\sum_{l=0}^{k}\frac{k!}{l!}\alpha_{i,k-1} - \sum_{j=1}^{s} k b_{ij} c_j^{k-1} - \sum_{j=1}^{r} v_{ij}\alpha_{jk} k!\right)\frac{h^k}{k!} y^{(k)}(t_{n-1}) = O\left(h^{p+1}\right).$$

Now each coefficient (call them d_k and $\tilde{d}_k$, respectively for the two equations) of $\frac{h^k}{k!} y^{(k)}(t_{n-1})$ (for each k) may be set to 0 since they are all independent of h. Then we have

$$\sum_{k=0}^{p} d_k \frac{z^k}{k!} = 0,$$
$$\sum_{k=0}^{p} \tilde{d}_k \frac{z^k}{k!} = 0,$$

which leads to two new equations. The first is

$$\sum_{k=0}^{p} c_i^k \frac{z^k}{k!} - \sum_{k=0}^{p}\sum_{j=1}^{s} k a_{ij} c_j^{k-1}\frac{z^k}{k!} - \sum_{k=0}^{p}\sum_{j=1}^{r} u_{ij}\alpha_{jk} z^k = 0.$$

The first summation may be identified as the first p + 1 terms in the Taylor expansion of $e^{c_i z}$, leaving a difference that is of order $O(z^{p+1})$. Reversing the order of the summations, the third term may be readily identified as $-\sum_{j=1}^{r} u_{ij} w_j(z)$. Reversing the order of summation for the second term and factoring as appropriate, we have

$$\sum_{j=1}^{s} a_{ij} z \sum_{k=1}^{p} c_j^{k-1}\frac{z^{k-1}}{(k-1)!} = \sum_{j=1}^{s} a_{ij} z \sum_{k=0}^{p-1} c_j^k \frac{z^k}{k!}.$$

Here the sum over k may be identified as the first p terms in the Taylor expansion of $e^{c_j z}$, leaving a difference that is O(z^p). Then the second term is $-\sum_{j=1}^{s} a_{ij} z e^{c_j z} + O\left(z^{p+1}\right)$. Thus the first equation is equivalent to:

$$e^{c_i z} - \sum_{j=1}^{s} z a_{ij} e^{c_j z} - \sum_{j=1}^{r} u_{ij} w_j(z) = O\left(z^{p+1}\right),$$

or, in matrix form,

$$e^{cz} = zAe^{cz} + Uw(z) + O\left(z^{p+1}\right).$$

The second new equation is:

$$\sum_{k=0}^{p} \sum_{l=0}^{k} \frac{k!}{l!} \alpha_{i,k-l} \frac{z^k}{k!} - \sum_{k=0}^{p} \sum_{j=1}^{s} k b_{ij} c_j^{k-1} \frac{z^k}{k!} - \sum_{k=0}^{p} \sum_{j=1}^{r} v_{ij} \alpha_{jk} k! \frac{z^k}{k!} = 0.$$

We interchange the order of summation and find for the second term similarly as for the second term above that we have $-\sum_{j=1}^{s} b_{ij} z e^{c_j z} + O\left(z^{p+1}\right)$. For the third term our expression is similar to that for the third term in the first equation and we have $-\sum_{j=1}^{r} v_{ij} w_j(z)$. For the first term we may observe that it is $e^z w_i(z) + O(z^{p+1})$ by looking at $e^z w_i(z)$ as the product of the first p+1 terms in the Taylor expansion for e^z times the terms in $w_i(z)$. If terms of the same order in z are combined and terms of order in z higher than p are dropped the expressions may be seen to be identical. Then the second new equation may be rewritten as

$$e^z w_i(z) - \sum_{j=1}^{s} b_{ij} z e^{c_j z} - \sum_{j=1}^{r} v_{ij} w_j(z) = O\left(z^{p+1}\right),$$

or in matrix form,

$$e^z w(z) - zBe^{cz} - Ve^{cz} = O\left(z^{p+1}\right).$$

These are equivalent to the second equation in the theorem. On the other hand, these steps may be reversed to obtain the order and stage order conditions. ■

It turns out that a restriction on the coefficient matrix B provides assurance that order and stage order conditions are met, according to the following theorem derived by Butcher (Reference 1). The symbol e will be used throughout this report for the vector of the appropriate length for the context for which each element is 1.

Theorem 2. Let r = s = p, Ve = e. Then the DIMSIM

$$\begin{bmatrix} A & I \\ B & V \end{bmatrix}$$

is of order p and stage order q = p if and only if

$$B = B_0 - AB_1 - VB_2 + VA,$$

where

$$\begin{aligned} B_{0,i,j} &= \frac{\int_0^{1+c_i} \phi_j(x)dx}{\phi_j(c_j)}, \\ B_{1,i,j} &= \frac{\phi_j(1+c_i)}{\phi_j(c_j)}, \\ B_{2,i,j} &= \frac{\int_0^{c_i} \phi_j(x)dx}{\phi_j(c_j)}, \end{aligned} \tag{12}$$

and where

$$\phi_j(x) = \prod_{k \neq j} (x - c_k). \tag{13}$$

Proof: See Butcher (Reference 1).

Note that using this theorem eliminates the elements of B as free parameters when deriving methods. It also has the following immediate corollary, since for any specified

vector c and matrices A and V a construction may be completed, leaving stability as the principal issue in deriving new methods.

Corollary: For each integer $p > 1$, DIMSIMs of order p and stage order $q = p$ exist for $s = r = p$, $U = I$, where s is the number of internal stages and r is the length of the external stage vector.

In what follows we will restrict ourselves to DIMSIMs for which $s = r = p = q$ and $U = I$.

STABILITY AND CONSISTENCY PROPERTIES OF DIMSIMS

We may note that in order to handle the simple scalar equation $y' = 0$ we must have the preconsistency condition $Ve = e$. The eigenvalues of V detemine the power-boundedness of the method, required for zero-stability, and these must all have magnitudes not greater than 1, and those with magnitude equal to one must have one-dimensional Jordan blocks. A typical design choice is to choose all eigenvalues to be 0 except for the unit eigenvalue associated with eigenvector e as is the case for Runge-Kutta methods, leaving V of rank 1 and all rows identical.

The standard consistency condition, related to the solution of the equation $y' = 1$, then requires that there must exist a consistency vector u such that $Be + Vu = e + u$ (Reference 11). We first note that equating the terms of order zero in z in the Taylor expansion about $z = 0$ for the first equation of Equation 11 tells us that $\alpha_{0j} = 1$, $j = 1,\ldots,p$, so that $\alpha_0 = e$ (α_0 is the first column of W). It follows that $Ve = e$ and $Be + V\alpha_1 = e + \alpha_1$ where α_1 is the second column of W, made up of elements α_{1j}, as may be seen by equating zeroth and first order terms in z (respectively) in the Taylor expansion about $z = 0$ for the second equation of Equation 11 and hence α_1 is a consistency vector for the method.

The stability matrix for a DIMSIM is (Reference 1)

$$M(z) = V + zB(I - zA)^{-1}. \tag{14}$$

This is easily seen from the method (Equation 2) using the standard test problem $y' = \lambda y$, $y(t_0) = y_0$. We solve the first equation in 2 for Y and, noting that $F(Y) = \lambda Y$, we obtain, setting $z = h\lambda$,

$$Y = (I - zA)^{-1} y^{[n-1]}.$$

Then we may substitute this expression in the second equation of 2 to obtain

$$y^{[n]} = \left(zB(I - zA)^{-1} + V\right)y^{[n-1]}.$$

Thus the region of absolute stability of a DIMSIM is the region

$$S = \left\{z : w \in \sigma\left(M(z)\right) \Rightarrow w < 1\right\}.$$

If S includes the entire open left half plane the method is called A-stable. We also define the associated stability polynomial

$$p(\omega, z) = \det\left(wI - M(z)\right).$$

A method may typically be verified to be A-stable either by using the Schur criterion (see, for example, Lambert (Reference 13)), or by reducing the stability polynomial to a familiar form associated with a Runge-Kutta method known to be A-stable.

A FEW SIMPLE EXAMPLES OF DIMSIMS

Butcher divides DIMSIMs into 4 categories, or types, using criteria of explicit or implicit and suitable or not suitable for parallel evaluation of stages. Note that for a matrix A that is lower triangular as is required for DIMSIMs, if it is also diagonal (including the zero matrix) the stage evaluations are completely independent of one another and may be carried out in parallel. Thus we have the following order 2 examples with c = [0,1], first developed by Butcher and illustrating his taxonomy (Reference 1):

<u>Type 1 (explicit, serial)</u>:

$$\begin{bmatrix} A & U \\ B & V \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ \frac{5}{4} & \frac{1}{4} & \frac{1}{2} & \frac{1}{2} \\ \frac{3}{4} & -\frac{1}{4} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{15}$$

This method has by design the same stability function (and region) as a 2-stage explicit Runge-Kutta methods of order 2. That is, the eigenvalues of M(z) are zero and R(z) where

the stability function of a Runge-Kutta method of order 2.

Type 2 (implicit, serial):

$$\begin{bmatrix} A & U \\ B & V \end{bmatrix} = \begin{bmatrix} \frac{2-\sqrt{2}}{2} & 0 & 1 & 0 \\ \frac{6+2\sqrt{2}}{7} & \frac{2-\sqrt{2}}{2} & 0 & 1 \\ \frac{73-34\sqrt{2}}{28} & \frac{-5+4\sqrt{2}}{4} & \frac{3-\sqrt{2}}{2} & \frac{-1+\sqrt{2}}{2} \\ \frac{87-48\sqrt{2}}{28} & \frac{-45+34\sqrt{2}}{28} & \frac{3-\sqrt{2}}{2} & \frac{-1+\sqrt{2}}{2} \end{bmatrix} \tag{16}$$

This method was designed to have a stability matrix with eigenvalues zero and R(z), where

$$R(z) = \frac{z\left(\sqrt{2}-1\right)}{z^2\left(\frac{3}{2}-\sqrt{2}\right) - z\left(2-\sqrt{2}\right)+1},$$

the same stability function as a 2-stage Singly-Diagonally Implicit Runge-Kutta (SDIRK) method of order 2 which is known to be A-stable. Therefore this method (Equation 16) is then itself A-stable.

Type 3 (explicit, parallel):

$$\begin{bmatrix} A & U \\ B & V \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{3}{8} & -\frac{3}{8} & -\frac{3}{4} & \frac{7}{4} \\ -\frac{7}{8} & \frac{9}{8} & -\frac{3}{4} & \frac{7}{4} \end{bmatrix} \tag{17}$$

This method has a stability polynomial

$$p(w,z) = w^2 - \left(1+\frac{3z}{4}\right)w - \frac{z}{4}(3z+1),$$

which yields a stability interval along the real axis of $\left[-\frac{4}{3},0\right]$.

Type 4 (implicit, parallel):

$$\begin{bmatrix} A & U \\ B & V \end{bmatrix} = \begin{bmatrix} \frac{3-\sqrt{3}}{2} & 0 & 1 & 0 \\ 0 & \frac{3-\sqrt{3}}{2} & 0 & 1 \\ \frac{18-11\sqrt{3}}{4} & \frac{-12+7\sqrt{3}}{4} & \frac{3-2\sqrt{3}}{2} & \frac{-1+2\sqrt{3}}{2} \\ \frac{22-13\sqrt{3}}{4} & \frac{-12+9\sqrt{3}}{4} & \frac{3-2\sqrt{3}}{2} & \frac{-1+2\sqrt{3}}{2} \end{bmatrix} \tag{18}$$

This method has the stability polynomial

$$p(w,z) = w^2 + \frac{2+\left(\sqrt{3}-3\right)z}{-2+\left(6-2\sqrt{3}\right)z+\left(3\sqrt{3}-6\right)z^2} w + z\frac{2-2\sqrt{3}+\left(8\sqrt{3}-12\right)z+\left(15-9\sqrt{3}\right)z^2}{\left(-2+\left(6-2\sqrt{3}\right)z+\left(3\sqrt{3}-6\right)z^2\right)^2}$$

and is found, using the Schur criterion, to be A-stable.

We may note by examining Equation 2 that for the serial DIMSIM types, the first internal stage must be calculated before the second stage, etc. For the parallel types, on the other hand, the calculations of the internal stages are independent of each other and could conceivably be performed on separate processors of a parallel computer. This is an example of "parallelism across the method," described by Gear in an early report (Reference 13) in which this form of parallelism was distinguished from "parallelism across the system" in which different equations or subsystems might be handled with different processors. Waveform relaxation (see, for example, Reference 14 and Reference 15) is an example of the use of this latter form of parallelism and it is possible that both types of parallelism could be combined in solving a single problem. It is evident that the parallelism across the method indicated here for DIMSIMs would only enable effective use of a number of processors equal to the number of internal stages while parallelism across the system could employ many processors in solving a large system. It may also be noted that the parallel types have fewer parameters and it would then be expected to be more difficult to find methods with desirable stability properties and other characteristics.

The type 3 methods are interesting in that they do not call for the calculation of internal stages, the external stages are the same as the internal stages. Also since A = 0, the first equation of Equation 11 implies that the external stages are the first p + 1 terms of the Taylor expansions of y at the stage points $t_{n-1} + c_i h$, since $e^{cz} = w(z) + O\left(z^{p+1}\right)$. The methods then use a Taylor series starting method and time marching is carried out using the equation

$$y^{[n]} = hBF\left(y^{[n-1]}\right) + Vy^{[n-1]}. \tag{19}$$

For example, with p = 2, $c_1 = 0$, $c_2 = 1$, we find (using methods described in the next section) that we may also derive the method

$$\begin{bmatrix} A & U \\ B & V \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ \frac{-1}{2} & \frac{3}{2} & 0 & 1 \end{bmatrix}. \tag{20}$$

Upon examining the time marching process of Equation 19 for this particular method, we see that it is equivalent to the the familiar second order Adams-Bashforth method,

$$y_n = y_{n-1} + \tfrac{h}{2}\left(3f_{n-1} - f_{n-2}\right).$$

For Type 3 methods the stage values approximate the solution at the stage points, since external stages and internal stages are equal here and the stage order condition requires that the solution be approximated by the internal stages at the stage points. These are each calculated using an explicit linear multistep method depending on previous values only from the last interval, with the formula for the ith component skipping the i - 1 last calculated solution points. A DIMSIM uses an interpolant then based on solution and derivative values from the stage points to produce final approximations at arbitrary points. Nevertheless the Type 3 time marching process can be seen as an example of an explicit cyclic linear multistep method (Reference 16) when stage points are evenly spaced, and only a slight modification would seem necessary for irregular spacing. For example, the ith element of the external stage vector approximates the solution at $t_{n-1} + c_i h$ and is given by

$$y_i^{[n]} = h\sum_{j=1}^{p} b_{ij} f\left(t_{n-2} + c_j h, y_j^{[n-1]}\right) + \sum_{j=1}^{p} v_j y_j^{[n-1]},$$

and for constant step size and uniform spacing of c from 0 to 1 (c_j = (j-1)/(p-1)) and recognizing that we really have a solution at each of the stage points, this becomes for the ith stage of the nth step, using the simplified notation y^k to the refer to the kth stage, numbering from the first stage of the first step,

$$y^{p(n-1)+i} = h\sum_{j=1}^{p} b_{ij} f\left(t_0 + \left(p(n-2) + j\right)h, y^{p(n-2)+j}\right) + \sum_{j=1}^{p} v_j y^{p(n-2)+j},$$

and this is in the familiar explicit linear multistep class of formulas. Some interesting features of this family when seen in this form include the use of the same combination of solution values for each element within a cycle ($v^T y^{[n-1]}$) and the omission of several previous values and derivatives in the calculation at most of the stage points. Clearly this solution process can utilize separate parallel CPUs for the p derivative evaluations required at each step.

APPROXIMATING THE SOLUTION AND THE NORDSIECK VECTOR

Nordsieck techniques are used extensively in the development and implementation of DIMSIMs. These use a vector of derivatives scaled with the step size and usually also with a factor of 1/(j - 1)! where j is the component number of the vector. Here we omit the exra factor, which can be readily provided through multiplication by a constant diagonal matrix Q = diag(1/0!, 1/1!, 1/2!, ..., 1/p!), and use the term "Nordsieck vector" to refer to a closely related vector that frequently appears here. We define the Nordsieck vector of length p + 1 as

$$\tilde{y}(x_n) = \begin{bmatrix} y(x_n) \\ hy'(x_n) \\ \vdots \\ h^p y^{(p)}(x_n) \end{bmatrix}. \tag{21}$$

Nordsieck (References 17 and 18) described a family of linear multistep methods using the vector $Q\tilde{y}(x_n)$ which provided an especially convenient approach to step-size changes. Nordsieck noted that solutions of ordinary differential equations could be reduced to finding interpolation polynomials to use in representing the solution, and his vector provided a readily scalable representation. These techniques have frequently been incorporated in implementing linear multistep methods and Butcher and Jackiewicz (Reference 3) have shown how they may be utilized effectively with DIMSIMs as well. Trocogna (Reference 19) has since then used this approach in implementing two-step Runge-Kutta methods (Reference 20).

Two matrices are defined which are used to relate the Nordsieck vector to the internal and external stages of the method. Let $F\left(Y^{[n]}\right)$ be a vector with kth component $f\left(Y_k^{[n]}\right)$. Then we can find matrices $\tilde{B}$ and $\tilde{V}$ such that

$$\tilde{y}(x_n) = h\tilde{B}F\left(Y^{[n]}\right) + \tilde{V}y^{[n-1]} + O\left(h^{p+1}\right). \tag{22}$$

These matrices can be calculated using the following theorem, announced in Butcher (Reference 1) and proven rigorously in Butcher and Jackiewicz (Reference 3). We first define $\tilde{z}$ as a vector of length p + 1,

$$\tilde{z} = \begin{bmatrix} 1 \\ z \\ \vdots \\ z^p \end{bmatrix}. \tag{23}$$

Theorem 3 (Butcher and Jackiewicz, Reference 3)**:** Assume that the method in Equation 2 has order p and stage order q = p or q = p - 1. Then the approximations in Equation 22 are correct to $O\left(h^{p+1}\right)$ if and only if

$$e^z \tilde{z} = z\tilde{B}e^{cz} + \tilde{V}w(z) + O\left(z^{p+1}\right). \tag{24}$$

Proof: The proof was developed by Butcher and Jackiewicz and is reproduced here for the sake of completeness. Define a matrix T such that

$$T = \begin{bmatrix} t_0 & t_1 & \cdots & t_p \end{bmatrix} = \begin{bmatrix} 1 & -1 & \frac{1}{2} & \cdots & \frac{(-1)^p}{p!} \\ 0 & 1 & -1 & \cdots & \frac{(-1)^{p-1}}{(p-1)!} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}.$$

Taylor expansions, the stage order and the problem definition in Equation 1 are used to obtain the relationships

$$\begin{bmatrix} y(x_n) \\ hy'(x_n) \\ \vdots \\ h^p y^{(p)}(x_n) \end{bmatrix} = \begin{bmatrix} t_0, t_1, \ldots, t_p \end{bmatrix} \begin{bmatrix} y(x_{n-1}) \\ hy'(x_{n-1}) \\ \vdots \\ h^p y^{(p)}(x_{n-1}) \end{bmatrix} + O\left(h^{p+1}\right),$$

and

$$\begin{bmatrix} hF(Y_1) \\ hF(Y_2) \\ \vdots \\ hF(Y_p) \end{bmatrix} = \begin{bmatrix} 0, e, c, \ldots, \dfrac{c^q}{q!} \end{bmatrix} \begin{bmatrix} y(x_{n-1}) \\ hy'(x_{n-1}) \\ \vdots \\ h^{q+1} y^{(q+1)}(x_{n-1}) \end{bmatrix} + O\left(h^{q+2}\right).$$

We also use Equation 3 to write

$$\begin{bmatrix} y_1^{[n-1]} \\ y_2^{[n-1]} \\ \vdots \\ y_p^{[n-1]} \end{bmatrix} = \left[\alpha_0, \alpha_1, \ldots, \alpha_p\right] \begin{bmatrix} y(x_{n-1}) \\ hy'(x_{n-1}) \\ \vdots \\ h^p y^{(p)}(x_{n-1}) \end{bmatrix} + O\left(h^{p+1}\right).$$

We now substitute these equations into Equation 22 to obtain

$$\sum_{i=0}^{p} t_i y^{(i)}(x_{n-1}) h^i = \sum_{i=1}^{q+1} \frac{\tilde{B} c^{i-1}}{(i-1)!} y^{(i)}(x_{n-1}) h^i + \sum_{i=0}^{p} \tilde{V} \alpha_i y^{(i)}(x_{n-1}) h^i + O\left(h^{p+1}\right) + O\left(h^{q+2}\right).$$

Since q = p or q = p - 1 we can combine terms to obtain

$$\left(\tilde{V}\alpha_0 - t_0\right) y(x_{n-1}) + \sum_{i=1}^{p} \left(\tilde{V}\alpha_i + \frac{\tilde{B} c^{i-1}}{(i-1)!} - t_i \right) y^{(i)}(x_{n-1}) h^i = O\left(h^{p+1}\right).$$

Equating terms of the same power in h we obtain

$$t_0 = \tilde{V}\alpha_0,$$

and

$$t_k = \tilde{B} \frac{c^{k-1}}{(k-1)!} + \tilde{V}\alpha_k, \quad k = 1, 2, \ldots, p.$$

Multiplying the kth equation by z^k and adding yields

$$\sum_{i=0}^{p} t_i z^i = z\tilde{B}\sum_{i=1}^{p}\frac{c^{i-1}z^{i-1}}{(i-1)!} + \tilde{V}\sum_{i=0}^{p}\alpha_i z^i,$$

which may be seen to equivalent to Equation 29 upon expansion of the exponentials in Taylor series. ■

Noting that

$$\begin{aligned} y^{[n]} &= W\tilde{y}(x_n) + O\left(h^{p+1}\right) \\ &= W\tilde{B}hF\left(Y^{[n]}\right) + W\tilde{V}y^{[n-1]} + O\left(h^{p+1}\right) \\ &= BhF\left(Y^{[n]}\right) + Vy^{[n-1]}, \end{aligned}$$

and comparing the corresponding terms we obtain the following so-called compatibility conditions

$$\begin{aligned} W\tilde{B} &= B, \\ W\tilde{V} &= V. \end{aligned} \tag{25}$$

IMPLEMENTING DIMSIMS

IMPLEMENTATION ISSUES

In order for an ODE solution method to be useful, certain capabilities must be provided. A numerical method will most effectively be applied using adaptive step-size selection based on an error tolerance. Thus neither too much work is done due to steps that are too short, nor is required accuracy sacrificed by using steps that are too long. This requires an ability to change step size, and also the ability to estimate error and suitable step length. Order changing is also desirable for maximum efficiency, but techniques for accomplishing this with DIMSIMs have not yet been developed. Furthermore, although typically the ODE solution is desired at certain output points, often evenly spaced, the ideal combination of efficiency and accuracy calls for integration steps to be as long as accuracy will allow. Thus interpolation should be used to obtain output values. Some sort of starting technique is required for the first step. Finally, for methods for which $c_1 = 0$ and $c_p = 1$, a significant savings in work is possible by using a new approach that will be described for evaluation of the first internal stage.

CHANGING STEP SIZE WITH DIMSIMS

We must introduce some notation to discuss the rescaling that becomes necessary to continue the calculation with a modified step size. We begin by calculating $y^{[1]}$ using $Y^{[1]}$ and $h_1 = t_1 - t_0$, and we assume now that $y^{[n-1]}$ has been calculated using $Y^{[n-1]}$ and $h_{n-1} = t_{n-1} - t_{n-2}$. We now calculate a $y^{[n]}$ corresponding to t_n using $Y^{[n]}$ and step size $h_n = t_n - t_{n-1}$. We denote

$$\tilde{y}(t_{n-1}) = \begin{bmatrix} y(t_{n-1}) \\ h_{n-1} y'(t_{n-1}) \\ \vdots \\ h_{n-1}^{p} y^{(p)}(t_{n-1}) \end{bmatrix}, \tag{26}$$

$$\tilde{\tilde{y}}(t_{n-1}) = \begin{bmatrix} y(t_{n-1}) \\ h_n y'(t_{n-1}) \\ \vdots \\ h_n^{p} y^{(p)}(t_{n-1}) \end{bmatrix}. \tag{27}$$

We also define the diagonal matrix

$$D = diag\left(1, \delta, \delta^2, \ldots, \delta^p\right), \tag{28}$$

where $\delta = \delta_n = \dfrac{h_n}{h_{n-1}}$. Furthermore we distinguish $y^{[n-1]}$ and $\hat{y}^{[n-1]}$, where $\hat{y}^{[n-1]}$ is rescaled to reflect a next step of h_n. This rescaling process will be described below. We desire to have $\hat{y}^{[n-1]}$ approximate $W\tilde{\tilde{y}}(t_{n-1})$ in the same way that $y^{[n-1]}$ approximates $W\tilde{y}(t_{n-1})$. This leads to the relations

$$y^{[n-1]} = W\bar{y}(t_{n-1}) + O\left(h_{n-1}^{p+1}\right),$$

$$\bar{y}(t_{n-1}) = h_{n-1}\tilde{B}F\left(Y^{[n-1]}\right) + \tilde{V}\hat{y}^{[n-2]} + O\left(h_{n-1}^{p+1}\right), \quad \hat{y}^{[0]} = y^{[0]},$$

$$W\tilde{\bar{y}}(t_{n-1}) = WD\bar{y}(t_{n-1}) = \hat{y}^{[n-1]} + O\left(h_n^{p+1}\right).$$

This indicates, after using the expression for the Nordsieck vector at t_{n-1}, that we can rescale by using the formula

$$\hat{y}^{[n-1]} = h_{n-1}WD\tilde{B}F\left(Y^{[n-1]}\right) + WD\tilde{V}\hat{y}^{[n-2]}. \tag{29}$$

We now have a modified numerical process, as follows:

$$\begin{aligned} Y^{[n]} &= h_n AF\left(Y^{[n]}\right) + \hat{y}^{[n-1]}, \\ \hat{y}^{[n]} &= h_n WD\tilde{B}F\left(Y^{[n]}\right) + WD\tilde{V}\hat{y}^{[n-1]}. \end{aligned} \tag{30}$$

We note (Reference 1) that zero-stability for a step-changing solver will then be determined by the eigenvalues of the matrix $WD\tilde{V}$. Thus any free parameters in determining $\tilde{B}$ and $\tilde{V}$ might well be used to ensure good stability for step-size changing. That this is always possible will now be shown.

Computation of the rescaling matrices is simplified, first of all, with all type 1 DIMSIMs with $c_1 = 0$. The equations to be satisfied include Equation 25 (the compatibility condition), Equation 24 and the condition that step-size changing zero stability be nonrestrictive, met where the matrix $WD\tilde{V}$ has only the one nonzero eigenvector e with the associated eigenvalue of 1. We are able to meet both of these conditions for a choice of $\tilde{V}$ such that the first row is v^T and all the other rows are 0, and for a choice of $\tilde{B}$ such that the first row is the same as the first row of B and the other elements are computed uniquely from a linear system. We quickly note that for $\tilde{V}$ of this form, $D\tilde{V}=\tilde{V}$, and $W\tilde{V}=V$ by the compatibility condition, and V is defined to be equal to ev^T. This indeed has the one nonzero eigenvector e with associated eigenvalue of 1. We now examine the condition 24 for the first rows of the two rescaling matrices,

$$e^z = z\tilde{B}_1 e^{cz} + \tilde{V}_1 w(z) + O\left(z^{p+1}\right).$$

We compare this with the similar condition in Equation 11 for the first rows of B and V, and noting on the left side the condition $c_1 = 0$, which makes $w_1(z) = 1$, we obtain

$$e^z = zB_1 e^{cz} + V_1 w(z) + O\left(z^{p+1}\right).$$

Clearly the first rows satisfy the same conditions, and hence the first rows of $\tilde{B}$ and $\tilde{V}$ can be taken the same as the first rows of B and V, respectively. We now consider the conditions on the remaining rows. If we take the remaining rows of $\tilde{V}$ to be 0, we then have the conditions for rows of $\tilde{B}$ as follows:

$$e^z z^{k-1} = z\tilde{B}_k e^{cz} + O\left(z^{p+1}\right), \ k = 2, \ldots, p+1.$$

We have p unknowns in each equation. But if we examine the number of conditions required to equate polynomial terms of degree 0 through p, we find that there will be no degree 0 condition, but degrees 1 through p will always appear. Thus we have p linear equations in p unknowns for each remaining row of $\tilde{B}$. Also, the matrices for each equation are the same but the right hand sides will always be different. And so long as the method is nonconfluent (that is, $c_i \neq c_j$ for $i \neq j$), there should be no problem in solving these systems, as may be observed from the formation of the matrix elements from the Taylor series. This is easily extended for both implicit and explicit DIMSIMs when c_1 is not 0. Taking $\tilde{V}$ as before, we find the situation with $\tilde{B}$ for rows after the first to be unchanged. The relationship of the equation for the first row to the equation for the first row of B is modified, however, since $w_1(z)$ is no longer 1. Recognizing that the the zero degree term in each component of w(z) is 1, we find that the equation

$$e^z - v^T w(z) = z\tilde{B}_1 e^{cz} + O\left(z^{p+1}\right)$$

has a zero term of degree 0, since the leading 1 in the Taylor expansion of e^z is cancelled by the leading 1 from the term $v^T e$. Thus we again have p equations in p unknowns. Note that the compatibility condition for $\tilde{B}$ must automatically be satisfied for this form of $\tilde{V}$ since multiplying Equation 24 by W and setting $\delta = 1$ produces the same equation that B must satisfy, the second relation of Equation 11. We summarize these observations in the following theorem which will be helpful in calculating the rescaling matrices $\tilde{B}$ and $\tilde{V}$.

Theorem 4: For nonconfluent DIMSIMs, we have

i) the rescaling matrix $\tilde{V}$ may be chosen to consist of a first row v^T and all other rows equal to 0;

ii) if $c_1=0$, the rescaling matrix $\tilde{B}$ will then have a first row that is identical to the first row of B,

$$e^z z^{k-1} = z\tilde{B}_k e^{cz} + O\left(z^{p+1}\right),\ k = 2,\ldots,p+1;$$

iii) if $c_1 \neq 0$, the first row of $\tilde{B}$ must satisfy the linear system

$$e^z - v^T w(z) = z\tilde{B}_1 e^{cz} + O\left(z^{p+1}\right);$$

iv) and for all choices of c_1, subsequent rows must each satisfy the linear system resulting from Taylor expansion of

$$e^z z^{k-1} = z\tilde{B}_k e^{cz} + O\left(z^{p+1}\right),\ k = 2,\ldots,p+1.$$

We see that the coefficients of $\tilde{B}$ for rows after the first are thus dependent only on the choice of stage points. It is evident that the higher derivatives of the Nordsieck vector are obtained in this approach by taking advantage of the high stage order. Applying the derivative function to the internal stages yields high order approximations to the solution derivatives at the stage points which yield linear systems for the higher derivatives through Taylor expansion .

Zero stability is obviously a rather mimimal condition. More careful step-size change stability requires examination of the stability matrix for this process. This may be written as

$$M(\delta) = zWD(\delta)\tilde{B}(I - zA)^{-1} + WD(\delta)\tilde{V} = zWD(\delta)\tilde{B}(I - zA)^{-1} + V \tag{31}$$

for the customary form of $\tilde{V}$ given by theorem 4. But it is not true that eigenvalues of this matrix will determine the growth of the external stage vector, since it is nonsymmetric and typically varies with each step, and this becomes a very difficult problem. However, lacking some better criterion, in a heuristic approach the eigenvalues of this matrix may be examined by calculating a sort of "pseudo-stability region" to provide some indication of the effect on stability of step-size changing with a view to aiding the determination for a solver of the bounds to set on step-size changes. This approach, pursued by Enenkel

(Reference 21) in his study of related general linear methods, yielded very restrictive results.

BUTCHER-JACKIEWICZ-TYPE INTERPOLATION

Butcher and Jackiewicz (Reference 3) proposed continuous interpolants of uniform order p of the form

$$\eta\left(t_{n-1}+\theta h_n\right)=h_n\beta_0(\theta)F\left(Y^{[n]}\right)+\gamma_0(\theta)\hat{y}^{[n-1]}. \tag{32}$$

Here we define $\beta_0(\theta)=\left[\beta_{01}(\theta),\beta_{02}(\theta),\ldots,\beta_{0s}(\theta)\right]$ and $\gamma_0(\theta)=\left[\gamma_{01}(\theta),\gamma_{02}(\theta),\ldots,\gamma_{0r}(\theta)\right]$, where the components are polynomials of degree p (or lower if certain coefficients become set to zero). For a Nordsieck interpolant we may write the interpolant in this form as well, and with the customary form for $\tilde{V}$, $\gamma_0(\theta)=v^T$. We further note that in order for compatibility with the equation for the first component of the Nordsieck vector, $\beta_0(1)$ and $\gamma_0(1)$ must be equal to the first rows of $\tilde{B}$ and $\tilde{V}$, respectively. This interpolant compatibility condition will be shown to be incorporated within the order condition provided in the following theorem derived by Butcher and Jackiewicz (Reference 3) in which the case q = p - 1, though not of particular interest here, is also included.

Theorem 5 (Butcher and Jackiewicz): If a DIMSIM has order p and stage order q = p or q = p - 1, then η approximates y with uniform order p if and only if

$$z\beta_0(\theta)e^{cz}+\gamma_0(\theta)w(z)=e^{\theta z}+O\left(z^{p+1}\right),\ \theta\in(0,1], \tag{33}$$

and w(z) is as defined above. Moreover, the interpolant η is continuous on the whole interval of integration if and only if

$$\begin{aligned}&\beta_0(0)=0,\\&\gamma_0(0)WD\tilde{B}=\beta_0(1),\\&\gamma_0(0)WD\tilde{V}=\gamma_0(1).\end{aligned} \tag{34}$$

Proof: Expanding $y^{[n]}$, $y(t_n)$, and $y'(t_{n+1}+ch)=hF(Y^{[n]})+O(h^{p+1})$ in Taylor series about t_{n-1} and assuming that the interpolant approximates y to uniform order p, we may write

$$\sum_{k=0}^{p}\frac{\theta^p}{k!}h_n^k y^{(k)}(t_{n-1}) = \beta_0(\theta)\sum_{k=1}^{q+1}\frac{c^{k-1}}{(k-1)!}h_n^k y^{(k)}(t_{n-1}) + \gamma_0(\theta)\sum_{k=0}^{p}\alpha_k h_n^k y^{(k)}(t_{n-1}) + O\left(h_n^{p+1}\right) + O\left(h_n^{q+2}\right),$$

where we denote the (i-1)th column of W as α_i. Since, as previously noted, the first column of W, α_0, is e, and q = p or q = p - 1, we can write

$$(\gamma_0(\theta)e - 1)y(x_{n-1}) + \sum_{k=1}^{p}\left(\frac{\beta_0(\theta)c^{k-1}}{(k-1)!} + \gamma_0(\theta)\alpha_k - \frac{\theta^k}{k!}\right)h^k y^{(k)}(x_{n-1}) = O\left(h^{p+1}\right).$$

We then generate p + 1 equations by setting coefficients of powers 0 to p of h_n to 0:

$$\gamma_0(\theta)e = 1,$$

$$\beta_0(\theta)\frac{c^{k-1}}{(k-1)!} + \gamma_0(\theta)\alpha_k = \frac{\theta^k}{k!}, \quad 1 \le k \le p.$$

We may now multiply each equation by an appropriate power of z and sum, obtaining an equation equivalent to the result we seek:

$$z\beta_0(\theta)\sum_{k=1}^{p}\frac{c^{k-1}z^{k-1}}{(k-1)!} + \gamma_0(\theta)\sum_{k=0}^{p}\alpha_k z^k = \sum_{k=0}^{p}\frac{\theta^k}{k!}.$$

These steps are reversible.

For continuity at node point x_n we must have $\eta(x_n -) = \eta(x_n +)$. The equation representing this relationship reads:

$$\begin{aligned} h_n\beta_0(1)f\left(Y^{[n]}\right) + \gamma_0(1)\tilde{y}^{[n-1]} &= h_{n+1}\beta_0(0)f\left(Y^{[n+1]}\right) + \gamma_0(0)\tilde{y}^{[n]} \\ &= h_{n+1}\beta_0(0)f\left(Y^{[n+1]}\right) + \gamma_0(0)\left(h_n WD\tilde{B}f\left(Y^{[n]}\right) + WD\tilde{V}\tilde{y}^{[n-1]}\right). \end{aligned}$$

Comparing coefficients of $f(Y^{[n+1]})$ we see that $\beta_0(0)=0$. Similarly, comparison of coefficients of $f(Y^{[n]})$and $y^{[n-1]}$ yields $\gamma_0(0)WD\tilde{B} = \beta_0(1)$ and $\gamma_0(0)WD\tilde{V} = \gamma_0(1)$, respectively. ■

The following corollary provides simplification of the conditions of this theorem.

Corollary: If a DIMSIM has order p and stage order q = p or q = p - 1, then η approximates y with uniform order p if and only if

$$z\beta_0(\theta)e^{cz} + \gamma_0(\theta)w(z) = e^{\theta z} + O\left(z^{p+1}\right),\ \theta \in (0,1], \tag{35}$$

and w(z) is as defined above. Moreover, the interpolant η is continuous on the whole interval of integration if and only if

$$\begin{aligned} &\beta_0(0) = 0, \\ &\gamma_0(0)B = \beta_0(1), \\ &\gamma_0(0)V = \gamma_0(1). \end{aligned} \tag{36}$$

Furthermore, if these conditions are met, the interpolant compatibility conditions that $\beta_0(1)$ and $\gamma_0(1)$ must be equal to the first rows of $\tilde{B}$ and $\tilde{V}$, respectively, will automatically be satisfied.

Proof: Looking again at the order condition in Equation 33, we find that because of continuity, we can now consider the case where θ is set to 0. Then since $\beta_0(0) = 0$, and $e^{\theta z} = 1$, we have $\gamma_0(0)w(z) = 1$. Now $w(z) = W\tilde{z}$. We can consider the various polynomial terms separately. Let $\hat{e}_1, \hat{e}_2$, etc., be the unit vectors with 1s in the appropriate position. Then $\gamma_0(0)W\hat{e}_1 = 1$, and $\gamma_0(0)W\hat{e}_k = 0$ for k > 1. Thus $\gamma_0(0)WD = [1 \quad 0 \quad \cdots \quad 0]$ regardless of the value of δ. Thus, from the order condition we have: $\gamma_0(0)WD\tilde{B} = \beta_0$ (the first row of $\tilde{B}$) and $\gamma_0(0)WD\tilde{V} = \gamma_0$ (the first row of $\tilde{V}$). Note that the interpolation coefficients will never depend on δ. Also note this eliminates the interpolant compatibility conditions as separate criteria; since there is no dependence on its value we may arbitrarily set δ = 1, in which case D = I. Then $WD\tilde{B} = WI\tilde{B} = W\tilde{B} = B$, and $WD\tilde{V} = WI\tilde{V} = W\tilde{V} = V$. ■

Deriving an interpolant is then a matter of finding coefficients to satisfy these relationships. In this report all interpolants of the form in Equation 32 and satisfying the conditions of Theorem 5 will be termed Butcher-Jackiewicz-type interpolants, while the interpolants proposed below by the author of this report will be termed Nordsieck or continuous Nordsieck interpolants.

Although examples of continuous interpolants of maximal order will be derived and utilized in the Implementing DIMSIMS section, we must note here that a continuous interpolant of maximal order in this form does not exist for all DIMSIMs. For example, we consider the type 2 example of Equation 16, shown again below:

$$\begin{bmatrix} A & U \\ B & V \end{bmatrix} = \begin{bmatrix} \frac{2-\sqrt{2}}{2} & 0 & 1 & 0 \\ \frac{6+2\sqrt{2}}{7} & \frac{2-\sqrt{2}}{2} & 0 & 1 \\ \frac{73-34\sqrt{2}}{28} & \frac{-5+4\sqrt{2}}{4} & \frac{3-\sqrt{2}}{2} & \frac{-1+\sqrt{2}}{2} \\ \frac{87-48\sqrt{2}}{28} & \frac{-45+34\sqrt{2}}{28} & \frac{3-\sqrt{2}}{2} & \frac{-1+\sqrt{2}}{2} \end{bmatrix},$$

with $c=[0,1]^T$. For this method the matrices $\tilde{B}$ and $\tilde{V}$ were calculated by Butcher (Reference 1) to be

$$\tilde{B} = \begin{bmatrix} \frac{-26+41\sqrt{2}}{28} & \frac{62-37\sqrt{2}}{28} \\ \frac{-48+51\sqrt{2}}{28} & \frac{64-33\sqrt{2}}{28} \\ \frac{-20+15\sqrt{2}}{14} & \frac{20-9\sqrt{2}}{14} \end{bmatrix},$$

and

$$\tilde{V} = \begin{bmatrix} \frac{-12+11\sqrt{2}}{14} & \frac{26-11\sqrt{2}}{14} \\ \frac{-30+3\sqrt{2}}{14} & \frac{30-3\sqrt{2}}{14} \\ -\frac{12+3\sqrt{2}}{7} & \frac{12+3\sqrt{2}}{7} \end{bmatrix}.$$

(Note: A separate calculation using the customary form for $\tilde{V}$ yielded the same result.)

We now apply the compatibility and continuity conditions (with the help of Mathematica) and show that an interpolant of the form described here cannot satisfy these. We look for vectors

$$\beta_0(\theta) = \left[\beta_{010} + \beta_{011}\theta + \beta_{012}\theta^2, \beta_{020} + \beta_{021}\theta + \beta_{022}\theta^2\right],$$

$$\gamma_0(\theta) = \left[\gamma_{010} + \gamma_{011}\theta + \gamma_{012}\theta^2, \gamma_{020} + \gamma_{021}\theta + \gamma_{022}\theta^2\right].$$

We immediately note from the first compatibility condition that β_{010} and β_{020} must be 0. We use the compatibility conditions to eliminate β_{012}, β_{022}, γ_{011}, and γ_{021}. We then use the second continuity condition to eliminate γ_{012} and γ_{022}. But then the third continuity condition, $\gamma_0(0)V=\gamma_0(1)$, becomes equivalent to

$$\left[\frac{1142683156482-1429914988166\sqrt{2}}{87418383556} \quad \frac{-6579484927094450-4651660909947086\sqrt{2}}{87418383556}\right]$$
$$=[0 \quad 0],$$

which contradicts the assumption that an interpolant of the given form exists.

A consequence of the fact that interpolants of the given form do not always exist is that a search for a suitable DIMSIM scheme should incorporate conditions for the existence of an interpolant of this desirable form. The existence of a suitable interpolant is crucial for the implementation of DIMSIMs in a waveform relaxation strategy, and for those methods for which this form does not exist, the Nordsieck form may always be used. A subsequent report will compare the performances of alternative interpolants for DIMSIM implementations for waveform relaxation.

We may readily obtain the following corollary to Theorem 5:

Corollary: If a continuous interpolant of the form of Equation 34 exists, then there exists a constant vector $\gamma_0(0)$ such that

i) $\gamma_0(0)y^{[n]} = y_n$ for each step of the integration process, where y_n is the approximation to the solution at t_n and is also given by the first component of the Nordsieck vector calculated using Equation 22.

ii) $\gamma_0(0)W = [1 \quad 0 \quad \cdots \quad 0]$.

iii) $\gamma_0(0)B = \tilde{B}_1$, $\gamma_0(0)V = \tilde{V}_1 = V_1 = v^T$, where the subscript 1 indicates the first row of the matrix and $V = ev^T$.

Proof: i) Setting θ to 0 in Equation 34 and using continuity at t_{n-1} and $\beta_0(0) = 0$, we obtain

$$\eta(t_{n-1}) = \gamma_0(0)y^{[n-1]} = \eta(t_{n-2} + h_{n-1}) = \tilde{y}_1(t_{n-1}).$$

We then note that we may simply adjust the subscript.

ii) See proof of Theorem 5.

iii) Since the first column of W is e, from ii), $\gamma_0(0)e = 1$. Then $\gamma_0(0)V = \gamma_0(0)ev^T = v^T$. Now see proof of Theorem 5. ■

NORDSIECK INTERPOLATION

The availability of the Nordsieck vector provides a ready interpolant. For any DIMSIM we may calculate coefficient matrices $\tilde{B}$ and $\tilde{V}$ using theorem 4, which will then yield Nordsieck vectors at each grid point. Interpolation can be carried out backward and extrapolation forward from a grid point using the Taylor expansion polynomial of degree p. Since the Nordsieck vector components are all locally accurate to $O(h^{p+1})$, the global accuracy of the interpolant is then $O(h^p)$, the same as the accuracy of the method at the grid points. This is provided for all DIMSIMs, including the Type 2 method above for which the Butcher-Jackiewicz-type interpolant does not exist. (Note that the availability of extrapolation accurate extrapolation forward is extremely useful in that it provides an excellent internal stage predictor for implicit methods. This will be developed more extensively in a report in preparation on implicit DIMSIMs.) If we have calculated an approximation to the Nordsieck vector at t_n as

$$\tilde{y}(t_n) \approx h_n \tilde{B} F\left(Y^{[n]}\right) + \tilde{V}\hat{y}^{[n-1]},$$

we may then carry out Nordsieck interpolatation at $t_{n-1} + \theta h_n$ using the Taylor series formula at the point t_n of the form

$$\eta\left(t_{n-1} + \theta h_n\right) = \begin{bmatrix} 1 & -1 & \frac{1}{2} & \cdots & \frac{(-1)^p}{p!} \end{bmatrix} D(1-\theta)\tilde{y}(t_n), \tag{37}$$

where

$$D(\delta) \equiv \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & \delta & 0 & \cdots & 0 \\ 0 & 0 & \delta^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \cdots \\ 0 & 0 & 0 & \cdots & \delta^p \end{bmatrix},$$

and where $1-\theta$ is used because interpolation is carried out to the left of the grid point, at t_n-$(1-\theta)h_n$. Alternatively interpolation to the right can be carried out. The formula changes

only slightly, with θ used instead of 1-θ and the negative signs in the multiplying vector are eliminated.

This interpolant has two drawbacks. First, since a fresh DIMSIM calculation of the Nordsieck vector is carried out at each grid point, these will be points of discontinuity. Secondly, Taylor series decrease in accuracy away from the node point. Both of these problems can be overcome, or at least mitigated, by increasing the Taylor series degree to p + 1. The extra component is calculated to exactly remove the discontinuity at the far grid point and will yield a 0 contribution at the grid point where the Nordsieck vector is calculated. For example, the interpolant above would be modified to produce a continuous Nordsieck interpolant by the addition of the term

$$\left(\tilde{y}_1(t_{n-1}) - \begin{bmatrix} 1 & -1 & \frac{1}{2} & \cdots & \frac{(-1)^p}{p!} \end{bmatrix} \tilde{y}(t_n)\right)(1-\theta)^{p+1} h_n^{p+1}. \tag{38}$$

Since this term should be quite small due to the higher power of h_n, the overall behavior of the interpolant should not degrade, and in fact tests described below in the chapter, "Developing Explicit DIMSIM ODE Solvers," indicate that it performs quite well.

DIMSIM ERROR ESTIMATES

An error estimate is intended to approximate the local error in the solution of an initial value problem. That is, if a DIMSIM has produced a solution at grid point t_{n-1} of y_{n-1}, then the DIMSIM solution y_n at t_n is compared to the exact solution $y(t_n;t_{n-1},y_{n-1})$ of the problem

$$\begin{cases} y'(t) = f(t, y(t)) \\ y(t_{n-1}) = y_{n-1} \end{cases}, \tag{39}$$

and the local error at t_n which is to be estimated is defined to be

$$err_n = y_n - y(t_n;t_{n-1}, y_{n-1}). \tag{40}$$

Unlike linear multistep and Runge-Kutta methods, DIMSIMs do not typically produce an approximation to the solution at the grid points in the time marching process. The first component of the Nordsieck vector must be separately computed from the external and internal stages when an approximation to the solution is desired. Furthermore, the external stages are neighboring trajectories that are averaged in a special way to produce a solution and the next internal and external stage vectors. Finally, the Nordsieck vector components deviate essentially independently as the solution process proceeds. Thus after the first step, the kth component approximating $h_n^k y^{(k)}(t_n)$ does not in fact represent to $O(h^{p+1})$ the scaled

kth derivative of the solution of the ordinary differential equation y'(t) = f(t,y) passing through the point $y(t_n)$ given by the first component, but rather to $O(h^p)$. Thus error estimation may be expected to proceed differently from the approach in the older families of methods. However, the work of estimating local error is still to determine the amount by which the approximation to the solution at the end of a step deviates from the local solution of the initial value problem y'(t) = f(t,y) passing through the approximation to the solution obtained at the end of the previous step. And this has proven to be possible with DIMSIMs.

Although error estimates for other cases have been derived (Reference 3), we will restrict ourselves to the case p = q. The stage order condition requires that if the solution is sufficiently smooth, we may write

$$Y_i = y\left(t_{n-1} + c_i h\right) + \xi_i y^{(p+1)}\left(t_{n-1}\right)h^{p+1} + O\left(h^{p+2}\right), \tag{41}$$

where the middle term on the right is designated the principal part of the error for Y_i. We note here that we are using the function y to refer to the exact local solution described above. Although theoretical investigation of the effect of the accumulation of global solution error on the validity of this local stage order condition remains to be carried out, there is ample experimental evidence to support the conjecture that Equation 41 holds true, and that global errors do not reduce the order of the leading stage error term. We assume that this is the case in the following development, which otherwise closely follows the approach developed by Butcher and Jackiewicz (Reference 3), in which possible stage order reduction is taken into account.

We note that the following general definition for local discretization error of the external stages applies to all DIMSIMs. The idea is to identify what is to be called the local discretization error of the external stages with the term of order h^{p+1} in the difference between the exact external stages at t_n and the calculated external stages, assuming that an exact Nordsieck vector is used initially at t_{n-1} and that stage and order conditions are met.

Definition: The local discretization error $le_i(t_n)$ of the ith external stage $y_i^{[n]}$ of the method I(2.2) at the point t_n is given by

$$le_i\left(t_n\right) = \sum_{k=0}^{p}\alpha_{ik}y^{(k)}\left(t_n\right)h^k - h\sum_{k=1}^{s} b_{ik}f\left(t_{n-1} + c_k h, Y_k^{[n]}\right) - \sum_{j=1}^{r}\sum_{k=0}^{p} v_{ij}\alpha_{jk}y^{(k)}\left(t_{n-1}\right)h^k, \tag{42}$$

where

$$Y_k^{[n]} = h\sum_{j=1}^{s} a_{kj}f\left(t_{n-1} + c_j h, Y_j^{[n]}\right) + \sum_{j=1}^{r}\sum_{i=0}^{p} u_{kj}\alpha_{ji}y^{(i)}\left(t_{n-1}\right)h^i, \quad k = 1,2,\ldots,s. \tag{43}$$

For the case p = q = r = s, U = I, and assuming that the local condition for stage order holds, we make these substitutions in Equation 42 to obtain the simplified expression

$$\begin{aligned} le_i(t_n) = \sum_{k=0}^{p} \alpha_{ik} y^{(k)}(t_n) h^k - h \sum_{j=1}^{p} b_{ij} f\left(t_{n-1} + c_j h, y\left(t_{n-1} + c_j h\right) + \xi_j y^{(p+1)} h^{p+1}\right) \\ - \sum_{j=1}^{p} \sum_{k=0}^{p} v_{ij} \alpha_{jk} y^{(k)}(t_{n-1}) h^k +, \; i = 1, \ldots, p \end{aligned} \quad . (44)$$

In any case, the vector of values $le_i(t_n)$ we designate as the local discretization error of the external stages.

Theorem 6 (Butcher and Jackiewicz, Reference 3) The local discretization error $le(t_n)$ of the external stages of Equation 2 at the point t_n is given by

$$le(x_n) = \varphi_p y^{(p+1)}(x_{n-1}) h^{p+1} + O\left(h^{p+2}\right), \tag{45}$$

where

$$\varphi_p = \sum_{k=1}^{p+1} \frac{\alpha_{p+1-k}}{k!} - \frac{Bc^p}{p!}. \tag{46}$$

Proof: The proof is given for more general choices of p, q, r, and s in Butcher and Jackiewicz (Reference 3) and is reproduced here for this restricted case for the sake of completeness. We may use Taylor expansion to express

$$y^{(k)}(t_n) = \sum_{l=0}^{p+1-k} y^{(k+l)}(t_{n-1}) \frac{h^l}{l!} + O\left(h^{p+2-k}\right), k = 0, 1, \ldots, p+1.$$

We then substitute this expression into Equation 44 and obtain

$$\sum_{k=0}^{p} \sum_{l=0}^{p+1-k} \frac{\alpha_{ik}}{l!} y^{(k+l)}(t_{n-1}) h^{k+l} = h \sum_{j=1}^{s} b_{ij} y'\left(t_{n-1} + c_j h\right) +$$
$$\sum_{j=1}^{p} \sum_{k=0}^{p} v_{ij} \alpha_{jk} y^{(k)}(t_{n-1}) h^k + le_i(t_n) + O\left(h^{p+2}\right).$$

We now expand $y'(t_{n-1}+c_j h)$ about t_{n-1} to get

$$\sum_{k=0}^{p}\sum_{l=0}^{p+1-k}\frac{\alpha_{ik}}{l!}y^{(k+l)}(t_{n-1})h^{k+l}=h\sum_{j=1}^{p}\sum_{k=1}^{p+1}\frac{b_{ij}c_j^{k-1}}{(k-1)!}y^{(k)}(t_{n-1})h^{k}$$
$$+\sum_{j=1}^{p}\sum_{k=0}^{p}v_{ij}\alpha_{jk}y^{(k)}(t_{n-1})h^{k}+le_i(t_n)+O(h^{p+2}).$$

We now reorder the summation on the left side:

$$\sum_{k=0}^{p}\sum_{l=0}^{p+1-k}\frac{\alpha_{ik}}{l!}y^{(k+l)}(t_{n-1})h^{k+l}$$
$$=\sum_{k=0}^{p}\sum_{l=0}^{\nu}\frac{\alpha_{ik}}{(\nu-k)!}y^{(\nu)}(t_{n-1})h^{\nu}+\sum_{l=1}^{p+1}\frac{\alpha_{i,p+1-l}}{l!}y^{(p+1)}(t_{n-1})h^{p+1}$$
$$=\sum_{k=0}^{p}\left(\sum_{l=0}^{p-k}\frac{\alpha_{i,k-l}}{l!}\right)y^{(k)}(t_{n-1})h^{k}+\sum_{l=1}^{p+1}\frac{\alpha_{i,p+1-l}}{l!}y^{(p+1)}(t_{n-1})h^{p+1}.$$

Then, interchanging summation orders on the right, we obtain

$$\sum_{k=0}^{p}\left(\sum_{l=0}^{k}\frac{\alpha_{i,k-l}}{l!}\right)y^{(k)}(t_{n-1})h^{k}+\sum_{l=1}^{p+1}\frac{\alpha_{i,p+1-l}}{l!}y^{(p+1)}(t_{n-1})h^{p+1}$$
$$=\sum_{k=1}^{p+1}\sum_{j=1}^{p}\frac{b_{ij}c_j^{k-1}}{(k-1)!}y^{(k)}(t_{n-1})h^{k}+\sum_{k=0}^{p}\sum_{j=1}^{p}v_{ij}\alpha_{jk}y^{(k)}(t_{n-1})+le_i(t_{n-1})+O(h^{p+2}).$$

We now combine terms and rearrange to find that

$$\sum_{k=0}^{p}\left(\sum_{l=0}^{k}\frac{k!\,\alpha_{i,k-l}}{l!}-\sum_{j=1}^{p}kb_{ij}c_j^{k-1}-\sum_{j=1}^{p}k!\,v_{ij}\alpha_{jk}\right)y^{(k)}(x_{n-1})\frac{h^{k}}{k!}$$
$$+\left(\sum_{l=1}^{p+1}\frac{\alpha_{i,p+1-l}}{l!}-\sum_{j=1}^{p}\frac{b_{ij}c_j^{p}}{p!}\right)y^{(p+1)}(x_{n-1})h^{p+1}=le_i(x_{n-1})+O(h^{p+2}).$$

The order conditions ensure that terms of order less than p + 1 in h must vanish, and so we have the result that

$$le_i(t_n) = \left(\sum_{l=1}^{p+1} \frac{\alpha_{i,p+1-l}}{l!} - \sum_{j=1}^{p} \frac{b_{ij} c_j^p}{p!} \right) y^{(p+1)}(t_{n-1}) h^{p+1} + O\left(h^{p+1}\right). \qquad \blacksquare$$

The connection between errors in the external stages and errors in the solution values is not immediately obvious. However, the value of $v^T le(x_n)$ takes on a special significance. Here v^T is both the row vector (all identical) of V and the left eigenvector associated with eigenvalue 1 of V. This will be called the principal part of the local discretization error. Albrecht (Reference 22) showed that for a broad class of methods, this is the quantity that should be controlled to maintain accurate integration, and as will be demonstrated in the Techniques for Obtaining Values for DIMSIMs section, this choice leads to very satisfactory results.

The error estimate for fixed step sizes may be found, as demonstrated by Butcher and Jackiewicz (Reference 3), in the form

$$v^T \varphi_p y^{(p+1)}(t_{n-1}) h^{p+1} = h \beta^T F\left(Y^{[n]}\right) + \gamma^T y^{[n-1]} + O\left(h^{p+2}\right), \tag{47}$$

and for variable step sizes only the minor modifications shown below are needed. The error estimate is then a linear combination of terms aready computed and thus is essentially free of computational cost. The β and γ vectors are determined by the method and may be different for initial steps, constant step sizes, and varying step sizes (involving δ but reducing to the constant step size formula for $\delta = 1$). They may be computed as follows.

Define a matrix

$$G = \left[0, e, c, \ldots, \frac{c^p}{p!} \right], \tag{48}$$

and a matrix

$$\tilde{T} = \left[\tilde{t}_0, \tilde{t}_1, \ldots, \tilde{t}_{p+1}\right] = \begin{bmatrix} 1 & -1 & \frac{1}{2} & \cdots & \dfrac{(-1)^{p+1}}{(p+1)!} \\ 0 & 1 & -1 & \cdots & \dfrac{(-1)^p}{p!} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}. \tag{49}$$

Then the following theorem applies:

Theorem 7 (Butcher and Jackiewicz, Reference 3): If the fixed step method in Equation 2 has stage order q equal to the order p and V is a rank one matrix such that Ve = e, then $v^T le(x_n)$ can be estimated by the formula

$$v^T le(x_n) = h\beta^T F\left(Y^{[n]}\right) + \gamma^T y^{[n-1]} + O\left(h^{p+2}\right), \tag{50}$$

where, for every step after the first, the vectors β and γ satisfy the system of equations

$$\begin{aligned} &\gamma^T e = 0 \\ &\beta^T \frac{c^{i-1}}{(i-1)!} + \gamma^T BG\bar{t}_i = 0, \;\; i = 1,2,\ldots,p \;\; . \\ &\beta^T \frac{c^p}{p!} + \gamma^T BG\bar{t}_{p+1} = v^T \varphi_p. \end{aligned} \tag{51}$$

Proof: The following is based on an earlier proof of Butcher and Jackiewicz (Reference 3) and is included here for completeness and to indicate where their proof requires that the step number n be greater than 1.

The first equation of Equation 51 is the equation resulting from terms of O(1) in Equation 50, since, as we have already noted, the first column of W is e, and so each component of $y^{[n-1]}$ is $y(x_{n-1}) + O(h)$. Since each column of a rank 1 matrix is proportional to e and V is chosen to be rank 1, we must then also have $\gamma^T V = 0$. We now use Taylor expansion, the stage order, and the initial value problem Equation 1 to write

$$hF\left(Y^{[n]}\right) = G\begin{bmatrix} y(x_{n-1}) \\ hy'(x_{n-1}) \\ \vdots \\ h^{p+1}y^{(p+1)}(x_{n-1}) \end{bmatrix} + O\left(h^{p+2}\right)$$

We may also write

$$hF\left(Y^{[n-1]}\right) = G\begin{bmatrix} y(x_{n-2}) \\ hy'(x_{n-2}) \\ \vdots \\ h^{p+1}y^{(p+1)}(x_{n-2}) \end{bmatrix} + O\left(h^{p+2}\right),$$

but we note that this only makes sense for n > 1, that is, for steps after the first. We may also write the Taylor series relationship

$$\begin{bmatrix} y(x_{n-2}) \\ hy'(x_{n-2}) \\ \vdots \\ h^{p+1}y^{(p+1)}(x_{n-2}) \end{bmatrix} = \tilde{T}\begin{bmatrix} y(x_{n-1}) \\ hy'(x_{n-1}) \\ \vdots \\ h^{p+1}y^{(p+1)}(x_{n-1}) \end{bmatrix} + O\left(h^{p+2}\right),$$

which of course only applies to the first step if the domain of validity of the differential Equation 1 extends sufficiently far to the left of x_0. Finally we use the second equation of the method Equation 2 to write

$$y^{[n-1]} = hBF\left(Y^{[n-1]}\right) + Vy^{[n-2]},$$

which clearly is meaningless for the first step. We substitute these expressions into Equation 50, and using $\gamma^T V = 0$ and Theorem 6 we obtain

$$v^T \varphi_p h^{p+1} y^{(p+1)}(x_{n-1}) = \left(\beta^T G + \gamma^T BG\tilde{T}\right)\begin{bmatrix} y(x_{n-1}) \\ hy'(x_{n-1}) \\ \vdots \\ h^{p+1}y^{(p+1)}(x_{n-1}) \end{bmatrix} + O\left(h^{p+2}\right). \qquad (52)$$

But we have

$$\beta^T G + \gamma^T BG\tilde{T} = \left[0, \beta^T e + \gamma^T BG\tilde{t}_1, \ldots, \frac{\beta^T c^p}{p!} + \gamma^T BG_p \tilde{t}_{p+1} \right].$$

Substituting into Equation 52 and equating terms of the same degree in h, we obtain the conditions of Equation 51. ■

For the first step, the accuracy of the approximation used in generating the starting vector $y^{[0]}$ becomes important. Although the minimum requirement is that

$$y_i^{[0]} = \sum_{i=0}^{p} \alpha_{ik} y^{(k)}(t_0) h^k + O\left(h^{p+1}\right),$$

it is possible to obtain more accurate estimates for the higher derivatives of the solution so that the accuracy becomes $O(h^{p+2})$. The fixed-step error formulas derived by Butcher and Jackiewicz may then be replaced by formulas of similar form but with different coefficients.

Theorem 8 (Initial Step Error Estimate): If the solution y(x) to the problem in Equation 1 is sufficiently smooth and the starting vector is calculated by a method correct up to $O(h^{p+2})$, then the error in y_1, the approximation to $y(x_1)$ calculated using the method in Equation 2 is

$$lte = y(t_1) - y_1 = \left(\frac{1}{(p+1)!} - \sum_{j=1}^{p} \tilde{B}_{1j} \frac{c_j^p}{p!} \right) \left(h\beta^T F\left(Y^{[1]}\right) + \gamma^T y^{[0]} \right) + O\left(h^{p+2}\right), \qquad (53)$$

provided vectors β and γ meet the following conditions:

$$
\begin{aligned}
&1)\gamma^T e = 0\\
&2)\beta^T e + \gamma^T \alpha_1 = 0\\
&3)\sum_{j=1}^{p} \beta_j \frac{c_j^{k-1}}{(k-1)!} + \gamma^T \alpha_k = 0, k = 2,\ldots,p\\
&4)\sum_{j=1}^{p} \frac{\beta_j c_j^p}{p!} = 1
\end{aligned}
\tag{54}
$$

Proof: We wish to calculate the term of $O(h^{p+1})$ in the expression $y(t_1) - y_1$. We use a Taylor expansion for the true solution about t_0 and use the expression for the first component of the Nordsieck vector in computing y_1. Then we have

$$
\begin{aligned}
y(t_1) &= \sum_{i=0}^{p+1} \frac{y^{(i)}(t_0)}{i!} h^i + O\left(h^{p+2}\right),\\
y_1 &= h\tilde{B}_1 F\left(Y^{[1]}\right) + \tilde{V}_1 y^{[0]}
\end{aligned}
$$

We use the definition of the starting vector and the stage order condition to write

$$
\begin{aligned}
y_i^{[0]} &= \sum_{j=0}^{p} \alpha_{ij} y^{(j)}(t_0) h^j + O\left(h^{p+2}\right),\\
Y_i^{[1]} &= y(t_0 + c_i h) + O\left(h^{p+1}\right).
\end{aligned}
$$

Now

$$
\begin{aligned}
&F_i\left(Y^{[1]}\right) \equiv f\left(Y_i^{[1]}\right) = f\left(y(t_0 + c_i h) + O\left(h^{p+1}\right)\right) = f\left(y(t_0 + c_i h)\right) + O\left(h^{p+1}\right)\\
&= y'\left(t_0 + c_i h\right) + O\left(h^{p+1}\right) = \sum_{j=0}^{p} y^{(j+1)}(t_0) \frac{h^j}{j!} + O\left(h^{p+1}\right).
\end{aligned}
$$

Then the term in h of order p + 1 in $y(t_1) - y_1$ is the lte and may be readily found to be

$$lte = h^{p+1} y^{(p+1)}(t_0)\left(\frac{1}{(p+1)!} - \sum_{j=1}^{p} \tilde{B}_{1j}\frac{c_j^p}{p!}\right).$$

We wish now to find an estimate

$$h^{p+1} y^{(p+1)}(t_0) = h\beta^T F\left(Y^{[1]}\right) + \gamma^T y^{[0]} + O\left(h^{p+2}\right).$$

We use the same expressions for F and the external stages. This yields

$$h^{p+1} y^{(p+1)}(t_0) = \sum_{j=1}^{p} \beta_j \sum_{k=0}^{p} \frac{c_j^k}{k!} h^{k+1} y^{(k+1)}(t_0) + \sum_{j=1}^{p} \gamma_j \sum_{i=0}^{p} \alpha_{ji} y^{(i)}(t_0) h^i + O\left(h^{p+2}\right).$$

The first relationship comes from considering terms of order 0. The second relationship results from equating terms of order 1 and noting that if some $c_j = 0$, the term of order 1 is produced by multiplying the term of order 0 in the Taylor expansion by h and hence we use a convention where 1 appears in place of the apparent undefined 0^0 in the last formula. The third set of formulas comes from considering orders 2 through p, while the fourth formula comes from setting the coefficients of terms of order p + 1 on the right to one. ■

We note that we have p + 2 equations in either case to determine 2p variables. For p = 2 the solution is unique, while for higher orders there are additional free parameters. These may be used to accomplish other purposes, for example to avoid poles that might arise in the formula.

Error estimation for variable step implementations have also been developed. We follow here the Nordsieck approach of Butcher and Jackiewicz (Reference 3) but also note that an alternative formulation has been developed in a paper by Jackiewicz, Vermiglio and Zennaro (Reference 23). We define h_n as t_n-t_{n-1}, and $\delta = h_n/h_{n-1}$ and we seek vectors $\beta = \beta(\delta)$ and $\gamma = \gamma(\delta)$ such that

$$v^T \varphi_p h_n^{p+1} y^{(p+1)}(t_{n-1}) = h_n \beta^T(\delta) F\left(Y^{[n]}\right) + \gamma^T(\delta)\tilde{y}^{[n-1]} + O\left(h_n^{p+2}\right). \quad (55)$$

It should be noted in the following modification of a theorem by Butcher and Jackiewicz (Reference 3) that the error formula includes the effect of rescaling and that error estimation is not carried out by simply using a fixed step formula with a rescaled external stage vector as is done in interpolation. Also note that variable stepsize does not apply before the second step. Note that in the following the validity of the local stage order condition is assumed.

Theorem 9: Assume that the method in Equation 2 with p = q = r = s, U = I, is implemented in variable stepsize mode using the Nordsieck technique and that V is rank one, Ve = e. Then

$$v^T le(t_n) = h_n \beta^T(\delta) F\left(Y^{[n]}\right) + \gamma^T(\delta) \tilde{y}^{[n-1]} + O\left(h_n^{p+2}\right), \tag{56}$$

if $\beta = \beta(\delta)$ and $\gamma = \gamma(\delta)$ satisfy the system of equations

$$\begin{aligned}
&1)\, \gamma^T W D \tilde{V} = 0 \\
&2) \gamma^T e = 0 \\
&3) \left(\beta^T + \tfrac{1}{\delta} \gamma^T W D \tilde{B}\right) e = 0 \\
&4) \beta^T \frac{c^{i-1}}{(i-1)!} + \frac{1}{\delta^i} \gamma^T W D \tilde{B} G \tilde{t}_i = 0, \quad i = 2,3,\ldots,p \\
&5) \beta^T \frac{c^p}{p!} + \frac{1}{\delta^{p+1}} \gamma^T W D \tilde{B} G \tilde{t}_{p+1} = v^T \varphi_p.
\end{aligned} \tag{57}$$

For the frequently occurring case where the first row of $\tilde{V}$ is v^T and the other rows are 0, the first condition simplifies to $\gamma^T e = 0$, eliminating it as a separate condition.

Proof: We follow the proof of Butcher and Jackiewicz (Reference 3) with some modifications. We proceed as in the proof of Theorem 6. First, we use the stage order condition and Equation 48 to obtain through Taylor expansion,

$$h_n F\left(Y^{[n]}\right) = G \begin{bmatrix} y(t_{n-1}) \\ h_n y'(t_{n-1}) \\ \vdots \\ h_n^{p+1} y^{(p+1)}(t_{n-1}) \end{bmatrix} + O\left(h_n^{p+2}\right),$$

$$h_{n-1} F\left(Y^{[n-1]}\right) = G \begin{bmatrix} y(t_{n-2}) \\ h_{n-1} y'(t_{n-2}) \\ \vdots \\ h_{n-1}^{p+1} y^{(p+1)}(t_{n-2}) \end{bmatrix} + O\left(h_{n-1}^{p+2}\right).$$

Taylor expansion yields the relationship

$$\begin{bmatrix} y(t_{n-2}) \\ h_{n-1}y'(t_{n-2}) \\ \vdots \\ h_{n-1}^{p+1}y^{(p+1)}(t_{n-2}) \end{bmatrix} = \tilde{T}\begin{bmatrix} y(t_{n-1}) \\ h_{n-1}y'(t_{n-1}) \\ \vdots \\ h_{n-1}^{p+1}y^{(p+1)}(t_{n-1}) \end{bmatrix} + O\left(h_{n-1}^{p+2}\right),$$

and rescaling with $\tilde{D} \equiv diag\left(1, \frac{1}{\delta}, \frac{1}{\delta^2}, \cdots, \frac{1}{\delta^{p+1}}\right)$ yields the relationship

$$\begin{bmatrix} y(t_{n-1}) \\ h_{n-1}y'(t_{n-1}) \\ \vdots \\ h_{n-1}^{p+1}y^{(p+1)}(t_{n-1}) \end{bmatrix} = \tilde{D}\begin{bmatrix} y(t_{n-1}) \\ h_{n}y'(t_{n-1}) \\ \vdots \\ h_{n}^{p+1}y^{(p+1)}(t_{n-1}) \end{bmatrix}.$$

Substituting, we then obtain

$$h_{n-1}F\left(Y^{[n-1]}\right) = G\tilde{T}\tilde{D}\begin{bmatrix} y(t_{n-1}) \\ h_{n}y'(t_{n-1}) \\ \vdots \\ h_{n}^{p+1}y^{(p+1)}(t_{n-1}) \end{bmatrix} + O\left(h_{n}^{p+2}\right).$$

Furthermore, using the relationship in Equation 22 and the rescaling formula in Equation 29, we obtain

$$\tilde{y}^{[n-1]} = h_{n-1}WD\tilde{B}F\left(Y^{[n-1]}\right) + WD\tilde{V}\tilde{y}^{[n-2]}.$$

We now substitute into Equation 55 to obtain

$$v^T \varphi_p h_n^{p+1} y^{(p+1)}(t_{n-1}) = \left(\beta^T G + \gamma^T W D \tilde{B} G \tilde{T} \tilde{D}\right) \begin{bmatrix} y(t_{n-1}) \\ h_n y'(t_{n-1}) \\ \vdots \\ h_n^{p+1} y^{(p+1)}(t_{n-1}) \end{bmatrix}$$
$$+ \gamma^T W D \tilde{V} \tilde{y}^{[n-2]} + O\left(h_n^{p+2}\right).$$

Since fixed stepsize D and $\tilde{D}$ are identity matrices, we may note that this expression agrees with the comparable expression derived earlier for fixed stepsizes. However, the term $\gamma^T W D \tilde{V} \tilde{y}^{[n-2]}$ requires additional consideration. We note that the first column of G is 0. Then, examining the terms created in multiplying the Nordsieck vector by $\beta^T(\delta)G$, we find that there is no term remaining of order h^0. Similarly since $\tilde{D}$ is a diagonal rescaling matrix and $\tilde{T}$ is lower triangular, there is also no term of order h^0. Then, since $\tilde{y}^{[n-2]} = e y(t_{n-2}) + O(h_{n-1})$ we must have $\gamma^T W D \tilde{V} e = 0$. For the case where the first row of $\tilde{V}$ is v^T and the other rows are 0 we have $W D \tilde{V} e = e$ and the zeroth order condition reduces to the requirement that $\gamma^T(\delta) e = 0$, which also eliminates the entire term since $W D \tilde{V} = W \tilde{V} = V = e v^T$ for this case. However, for more general $\tilde{V}$ matrices it has been convenient to impose the sufficient condition $\gamma^T W D \tilde{V} = 0$.

We now use our knowledge of the nature of G in the first term and $\tilde{D}$ in the second to find that

$$\beta^T G + \gamma^T W D \tilde{B} G \tilde{T} \tilde{D} = \left[0 \quad \beta^T e + \tfrac{1}{\delta} \gamma^T W D \tilde{B} G \tilde{t}_1 \quad \cdots \quad \beta^T \frac{c^p}{p!} + \tfrac{1}{\delta^{p+1}} \gamma^T W D \tilde{B} G \tilde{t}_{p+1} \right]$$

We note that $G\tilde{t}_1 = e$. Thus the first order condition simplifies to $\left(\beta^T + \frac{1}{\delta} \gamma^T W D \tilde{B}\right) e = 0$. The other equations of Equation 37 then follow by equating terms of the same degree. ■

TECHNIQUES FOR OBTAINING STARTING VALUES FOR DIMSIMS

In general it is the derivatives that must be computed to obtain starting values. Techniques will be illustrated here for methods of second order that may be extended to methods of higher order. For a second order method only the second derivative becomes a problem, since the first derivative may be computed using the derivative function f of Equation 1. The derivative of the derivative function could be computed symbolically

whenever f is available in a symbolic form, but this will be too complicated for many functions of interest. Otherwise, an approximation for y'' may be calculated, which needs only to be correct to 1st order to provide satisfactory starting values. Only one additional solution point is needed, $y(x_0 + h_0)$, where h_0 is some small value (note that the first DIMSIM integration step size is designated h_1). The Taylor expansion at x_0 then yields a convenient expression with a first order error:

$$y(t_1) = y(t_0) + h_0 y'(t_0) + \tfrac{h_0^2}{2} y''(t_0) + O\left(h_0^3\right),$$

so we have the expression

$$y''(t_0) = \frac{2}{h_0}\left[\frac{y_1 - y_0}{h_0} - y_0'\right] + O(h_0). \tag{58}$$

Here we substitute the approximation

$$y_1 = y(t_1) + O\left(h_0^3\right),$$

where y_1 must be calculated using a second order Runge-Kutta method to provide a second derivative correct to $O(h_0)$. It may be noticed that an approximation for $y(t_1)$ is obtained as part of the starting procedure. However it is the external stages that are needed and they will be calculated at t_1 using the DIMSIM. New function evaluations will be required, even if the value of c_1 is 0 and h_0 is actually used as the initial step size.

Although adequate starting values are obtained with a first order estimate of the second derivate, as was shown in Theorem 8, in order to obtain a reliable error estimate for the first step for a second order DIMSIM, it is necessary to obtain an $O\left(h_0^2\right)$ approximation for $y''(t_0)$. This may be done using a 3rd order Runge-Kutta method and either two calculated points or one calculated point and a functional evaluation, as follows. Assume we calculate two points y_1 and y_2 from (t_0,y_0) using a 3rd order (explicit) method. Note that if both are calculated from the starting point there will be no stability problems, no subsequent steps are taken with the method. We can express both in terms of Taylor expansions of $y(t_0+h_1)$ and $y(x_0+h_2)$ to $O(h^4)$, since these are the accurate with error $O(h^4)$. Let $h_1 = h_0/2$ and $h_2 = h_0$. Then we have:

$$y_1 = y_0 + \frac{h_0}{2} y_0' + \frac{h_0^2}{8} y_0'' + \frac{h_0^3}{48} y_0''' + O\left(h_0^4\right)$$
$$y_2 = y_0 + h_0 y_0' + \frac{h_0^2}{2} y_0'' + \frac{h_0^3}{6} y_0''' + O\left(h_0^4\right).$$

We can eliminate the third derivative term between these two equations and come up with the following expression for the second derivative:

$$y_0'' = \frac{-2}{h_0^2}\left(y_2 - 8y_1 + 7y_0 + 3h_0 y_0'\right) + O\left(h_0^2\right). \tag{59}$$

Alternatively we can calculate a y_1 at t_0+h_0, do a function evaluation there to determine its first derivative y_1', and use Taylor expansions to develop a different formula, also correct to second order:

$$y_1 = y_0 + h_0 y_0' + \frac{h_0^2}{2} y_0'' + \frac{h_0^3}{6} y_0''' + O\left(h_0^4\right)$$
$$y_1' = y_0' + h_0 y_0'' + \frac{h_0^2}{2} y_0''' + O\left(h_0^3\right).$$

Multiplying through the second equation by h_0 we have an error term that is 4th order, the third derivative can again be eliminated and we arrive at the formula:

$$y_0'' = \frac{2}{h_0^2}\left(3\left(y_1 - y_0\right) - h_0\left(y_1' + 2y_0'\right)\right) + O\left(h_0^2\right). \tag{60}$$

Limited testing seemed to indicate that Equation 59 produced somewhat better results. However, the second method requires fewer function evaluations. But a great advantage to either approach is that they yield a convenient estimate for the third derivative, and this may be used to provide a very accurate a priori error estimate enabling optimal step size selection. This will be explained in more detail in the next section.

STEP SIZE SELECTION STRATEGY

The adaptive approach employed for each step is to use the error estimation techniques outlined above to obtain an estimate of the error generated with the step. If the error exceeds the tolerance the step size is halved, the external stage vector is rescaled using Equation 29, and the step is repeated. This continues until a result is obtained within tolerance or the maximum allowable number of attempts is exceeded, which terminates the integration with an error message. On the other hand, if a step is successful, a new step

size is calculated for the next step. This is done with the standard formula provided, for example, by Hairer, Norsett, and Wanner (Reference 18). The optimal step size, h_{opt}, will produce an error equal to the tolerance. The error is assumed for both the previous step and the next step to be of the form of the left hand side of Equation 55 with approximately the same derivative factor. Then we estimate

$$h_{opt} = h\left(\frac{1}{err}\right)^{\frac{1}{p+1}}.$$

There are some issues that must be considered in changing step size, even after a computation of optimal step size has been carried out. First, the amount of work that is necessary to complete an integration is minimized if the longest possible step sizes are used. On the other hand, failed steps are expensive since all the step calculations, including rescaling external stages, derivative evaluations, and error estimation must be repeated. Therefore, since error estimates are simply estimates, a safety factor is always utilized, usually chosen to be 0.8 or 0.9. Furthermore, large changes in optimal step-size calculations are an indication of rapid changes in the solution to the system being solved which lessens the value of error estimates, and so a maximum step-size increase ratio is set. Finally, error estimates and integration behavior deteriorate with frequent step-size changes, so it is desirable once a step size has increased to prohibit further increases for a few steps. There are then three parameters to be determined heuristically that can significantly affect the performance of a solver, the safety factor, the maximum step-size increase ratio, and the number of successive steps after a step-size change in which step size is kept from increasing.

Integration at the right end point may proceed in a few different ways. If the interval from the last mesh point to the end point is small, extrapolation may be used. Alternatively, the solver may integrate past the end point using the step length calculated from the error estimate and interpolate back. Finally, the right end point may be chosen as the final mesh point for the last integration step. This third alternative is used here because it is most appropriate for waveform relaxation. Extrapolation does not yield an interpolant for the final interval, which is the actual required output with waveform relaxation. Also, in waveform relaxation the derivative function is undefined past the end of the current window since interpolants for variables associated with other subsystems of the overall problem have not yet been determined. Thus the third alternative is the technique chosen for this work.

Choosing a suitable size for the first step has been almost as much of an art as a science and approaches are typically taken based, to a considerable extent, on heuristics. It is desirable for accuracy to choose a very small first step and then use the error estimator to determine step-size changes for subsequent steps. However, too small a first step will result in too many small steps as the integration progresses while the step size is increasing. Three different approaches were examined in developing DIMSIM solvers. Shampine and Gordon (Reference 24) used the following selection for initial step size:

$$h = \min\left(h_0, \frac{1}{4}\left|\frac{0.5*tol}{f(t_0, y_0)}\right| \right)^{\frac{1}{2}}. \tag{61}$$

Here the input h_0 value is a user-supplied estimate to prevent too large a value from being used, and tol is the specified tolerance. The choice comes from estimating the error of a first order method as h times the error of a zero order method, and then calculating a step size to produce an error of half the tolerance. This step size is then divided by 4 to produce a conservative value. Their approach is used in a linear multistep solver in which the first step is of first order, which renders it less suitable for a higher order DIMSIM solver.

A more sophisticated approach suitable for higher order solvers is presented by Hairer, Norsett and Wanner (Reference 18) and originally developed by Gladwell, Shampine, and Brankin (Reference 25). The local truncation error is assumed to be of the form

$$lte \approx Ch^{p+1}y^{(p+1)}(t_0). \tag{62}$$

They then recommend the following process. The preliminary value is the step size that produces an Euler step yielding a solution change of 1%. This is used to then obtain an estimate for the norm of the second derivative. The larger of the first and second derivatives is used as an estimate for $Cy^{(p+1)}(t_0)$, and a value of h is chosen to produce an lte of around 1%. Various threshholds are set to avoid bad choices for more exceptional cases. In summary, these steps are followed:

1. Let $d_0 = \|y_0\|$ and $d_1 = \|f(t_0, y_0)\|$, where $\|z\| = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{z_i}{sc_i}\right)^2}$ and sc is a vector of scale factors, here $sc_i = Atol_i + |y_{0i}| Rtol_i$ with Atol and Rtol vectors of absolute and relative error tolerances.

2. If d_0 or d_1 is less than 10^{-5}, set $h_0 = 10^{-6}$, otherwise let $h_0 = 0.01(d_0/d_1)$.

3. Let $y_1 = y_0 + h_0 f(t_0, y_0)$ and find $f(t_0 + h_0, y_1)$.

4. Let $d_2 = \frac{1}{h}\|f(t_0 + h_0, y_1) - f(t_0, y_0)\|$.

5. If $\max(d_1, d_2) \le 10^{-15}$ then set $h_1 = \max\left(10^{-6}, h_0 \cdot 10^{-3}\right)$, otherwise,

$$h_1 = \sqrt[p+1]{\frac{0.01}{\max(d_1, d_2)}}.$$

6. Use starting step size $h = \min(100h_0, h_1)$.

This was the first method that was tried for choosing the initial step size.

An alternative approach with considerable promise has been devised as part of this research effort. For a 5th order DIMSIM, for example, we would proceed as follows: Any small step may be used with an explicit 6th order Runge-Kutta integrator to produce 3 values for the solution at $t_0 + h_0$, $t_0 + 2h_0$, and $t_0 + 4h_0$. Using two of the three possible derivative function evaluations to produce values for the derivative, there are 5 equations in 5 unknowns, producing values for the $h_0^k y^{(k)}(t_0)$ for k ranging from 2 to 5 as needed for the starting vector, plus a value for $h_0^6 y^{(6)}(t_0)$, which may be used with the existing formula to provide an accurate a priori error estimate for the first step before the DIMSIM is even applied, using Equation 53. This may be used with the tolerance value to provide a simple formula for the appropriate step size for the first step. A safety factor of perhaps 2 should be provided based on heuristics from further tests. The starting vector calculated using h_0 would only need rescaling, prior to actual use. This will avoid the usual step size buildup at the beginning and improve accuracy. Furthermore, one might utilize alternative error formulas for solvers of different orders, with no additional derivative function evaluations required, to obtain an optimal order selection for the problem, at least in the region of the first step.

Some problems encountered with early testing of this approach indicate that it is important to utilize a suitable h_0 value. It should be noted that the most difficult value to calculate in the Nordsieck vector will be the term of $O(h_0^{p+1})$ if h_0 is small, because it is calculated as a linear combination of terms that may be near 1. Thus machine precision and roundoff error may become significant. The problem of machine precision is greatly simplified when a quadruple precision data type (real*16) is available. Furthermore, if h_0 is too large, the Runge-Kutta integration will not be sufficiently accurate. The estimate for a suitable value of h_0 may be obtained from the heuristic relationship

$$10 \cdot eps = scfac \cdot h_0^{p+1},$$

where eps is the machine epsilon and scfac is a scale factor reflecting the scaling of the problem. A reasonable choice of scfac might be 1 or the minimum of y_0 and y_0' with the maximum taken if one of the two is very small (an approximation to 0). This assumes that the (p + 1)st derivative is around the same size as scfac. The idea is to choose h_0 so that the smallest and most difficult term to calculate is on the order of ten times the machine epsilon. If h_0 is smaller, roundoff error will create problems, while if h_0 is too large the approximation accuracy will suffer. This process is more extensively illustrated as it is used for obtaining starting procedures in the Techniques for Obtaining Starting Values for DIMSIMs section.

ALTERNATIVE REPRESENTATIONS

The DIMSIM solution equation and the rescaling formula provide relationships among current external stage vectors, previous external stage vectors, and derivative vectors at current or previous internal stage points. These enable alternative representations of formulas for such things as error estimates, intepolants, Nordsieck vectors, and rescaling. These alternative formulas are mathematically completely equivalent but may have somewhat different roundoff properties. In some cases significant differences for waveform relaxation implementations in memory requirements and in message passing volume for parallel computing will result from the choice of representation, as will be described below. Alternative representations are also helpful in shortening recursions in deriving stability regions for predictor-corrector implementations, and this will be used in the report to follow on implicit DIMSIMs. The following relationships are not exhaustive but are provided as examples of useful forms.

We again write the second equation of Equation 2, using an alternative expression for V:

$$y^{[n]} = hBF\left(Y^{[n]}\right) + ev^T y^{[n-1]}. \tag{63}$$

We can then write, for nonsingular B, (implicitly assumed in this section wherever B^{-1} appears)

$$hF\left(Y^{[n]}\right) = B^{-1}y^{[n]} - B^{-1}ev^T y^{[n-1]}. \tag{64}$$

Alternatively, we may use the rescaling formula in Equation 29 to express

$$\hat{y}^{[n-1]} = h_{n-1}WD\tilde{B}F\left(Y^{[n-1]}\right) + WD\tilde{V}\hat{y}^{[n-2]}.$$

We can then apply these as follows:

a. Error Estimates: The fixed step formula has been expressed in Equation 50 as

$$v^T le(t_n) = h\beta^T F\left(Y^{[n]}\right) + \gamma^T y^{[n-1]} + O\left(h^{p+2}\right),$$

and we may produce alternative representations as described in the following.

Corollary to Theorem 7: If the fixed step method in Equation 2 has stage order q equal to the order p and V is a rank one matrix such that Ve = e, then $v^T le(x_n)$ can be estimated by Equation 50 and either of the formulas

$$v^T le(t_n) = h\beta_1^T F\left(Y^{[n]}\right) + h\beta_2^T F\left(Y^{[n-1]}\right) + O\left(h^{p+2}\right), \tag{65}$$

and

$$v^T le(t_n) = \gamma_1^T y^{[n]} + \gamma_2^T y^{[n-1]} + O\left(h^{p+2}\right), \tag{66}$$

where β and γ are given by the formulas of Equation 51, $\beta_1 = \beta$, $\beta_2^T = \gamma^T B$, and γ_1 and γ_2 are given by

$$\begin{aligned} \gamma_1^T &= \beta^T B^{-1}, \\ \gamma_2^T &= \gamma^T - \beta^T B^{-1} e v^T. \end{aligned} \tag{67}$$

Proof: To obtain Equation 66 we substitute Equation 64 into Equation 50 to obtain

$$v^T le(t_n) = \beta^T\left(B^{-1}y^{[n]} - B^{-1}ev^T y^{[n-1]}\right) + \gamma^T y^{[n-1]} + O\left(h^{p+2}\right),$$

and the result follows immediately. Equation 65 results from substituting Equation 63 for the previous step into Equation 50 to obtain

$$v^T le(t_n) = h\beta^T F\left(Y^{[n]}\right) + \gamma^T\left(hBF\left(Y^{[n-1]}\right) + ev^T y^{[n-2]}\right).$$

The extra term $\gamma^T e v^T y^{[n-2]}$ is 0 since $\gamma^T e=0$ from Equation 51. ■

We have similar relationships for variable step size. Note that δ refers to δ_n in the following.

Corollary to Theorem 9: If the variable step method in Equation 30 has stage order q equal to the order p and V is a rank one matrix such that Ve = e, then $v^T le(x_n)$ can be estimated by Equation 56 and any of the formulas

$$v^T le(t_n) = h_n \beta_1^T(\delta_n) F\left(Y^{[n]}\right) + h_{n-1}\beta_2^T(\delta_n, \delta_{n-1}) F\left(Y^{[n-1]}\right) + O\left(h^{p+2}\right), \tag{68}$$

and

$$v^T le(t_n) = \gamma_1^T(\delta) y^{[n]} + \gamma_2^T(\delta)\hat{y}^{[n-1]} + O\left(h^{p+2}\right), \tag{69}$$

where β and γ are given by the formulas of Equation 57, $\beta_1 = \beta$, $\beta_2 = \gamma^T(\delta_n) W D(\delta_{n-1})\tilde{B}$, with $\tilde{V}$ in standard form, and γ_1 and γ_2 are given by

$$\begin{aligned} \gamma_1^T &= \beta^T(\delta) B^{-1}, \\ \gamma_2^T &= \gamma^T(\delta) - \beta^T(\delta) B^{-1} e v^T. \end{aligned} \tag{70}$$

Proof: To obtain Equation 69 we substitute Equation 64 into Equation 56 to obtain

$$v^T le(t_n) = h\beta^T(\delta)\left(B^{-1} y^{[n]} - e v^T y^{[n-1]}\right) + \gamma^T(\delta) y^{[n-1]},$$

and the result follows from identification of coefficients. Equation 68 results from substituting Equation 29 into Equation 56 to obtain

$$v^T le(t_n) = h_n \beta^T(\delta_n) F\left(Y^{[n]}\right) + \gamma^T(\delta_n)\left(W D(\delta_{n-1})\tilde{B} F\left(Y^{[n-1]}\right) + W D \tilde{V}\hat{y}^{[n-2]}\right).$$

Assuming the standard form of $\tilde{V}$ with a first row of v^T and all the other rows identically 0, we may note that $WD\tilde{V} = V = ev^T$. This results in the extra term $\gamma^T(\delta) e v^T \hat{y}^{[n-2]}$, but $\gamma^T(\delta) e = 0$ from Equation 57. ■

A similar approach can be used for the first step error estimate.

b. Interpolants:

The most payoff from using alternative formulas can be expected here, since interpolants over some number of steps are stored and passed between parallel processors as they are used to represent solutions over a window in waveform relaxation. The

formula derived above requires storing and passing both external stage vectors for each step within a window and the vectors of derivatives at internal stage points, at each grid point requiring storage and passing of 2Mp numbers, where M is the number of equations in a subsystem and p is the order of the DIMSIM. The requirement for a Nordsieck interpolant is only M(p + 1), but in a representation using only external stage vectors this number becomes just Mp. We have the following alternative formulations. It should be noted carefully which external stage vectors are rescaled.

Corollary to Theorem 5: If a DIMSIM has order p and stage order q = p or q = p - 1, then a Butcher-Jackiewicz-type interpolant η given by either of two equivalent forms approximates y with uniform order p and is continuous on the whole interval of integration if and only if conditions of Equations 35 and 36 are met and the interpolant is given by Equation 34 or

$$\eta(t_{n-1}+\theta h_n)=\gamma_{0,1}(\theta)y^{[n]}+\gamma_{0,2}(\theta)\hat{y}^{[n-1]}, \tag{71}$$

where

$$\begin{aligned}\gamma_{0,1}(\theta)&=\beta_0(\theta)B^{-1},\\ \gamma_{0,2}(\theta)&=\gamma_0(\theta)-\beta_0(\theta)B^{-1}ev^T.\end{aligned} \tag{72}$$

Proof: This follows from identification of coefficients after direct substitution of Equation 64 into Equation 34 to obtain

$$\eta(t_{n-1}+\theta h_n)=\beta_0(\theta)\left(B^{-1}y^{[n]}-B^{-1}ev^T\hat{y}^{[n-1]}\right)+\gamma_0(\theta)\hat{y}^{[n-1]}. \qquad \blacksquare$$

For the Nordsieck vector we have the following.

Corollary to Theorem 3: If the method in Equation 2 satisfies Equation 24, then the Nordsieck vector $\tilde{y}$ at t_n may be approximated to $O(h^{p+1})$ using either the equivalent formula of Equation 22 or

$$\tilde{y}(t_n)=\tilde{V}_1y^{[n]}+\tilde{V}_2\hat{y}^{[n-1]}, \tag{73}$$

where

$$\tilde{V}_1 = \tilde{B}B^{-1},$$
$$\tilde{V}_2 = \tilde{V} - \tilde{B}B^{-1}ev^T. \tag{74}$$

Proof: This follows from identification of coefficients after direct substitution of Equation 64 into Equation 22 to obtain

$$\tilde{y}(t_n) = \tilde{B}\left(B^{-1}y^{[n]} - B^{-1}ev^T\hat{y}^{[n-1]}\right) + \tilde{V}\hat{y}^{[n-1]}. \quad \blacksquare$$

c. Rescaling: Since formulas used above include some rescaled and some unrescaled external stage vectors, the ability to rescale using only external stage vectors is important. We note that if we can rescale using Equation 29, we can also rescale using

$$\hat{y}^{[n-1]} = WD_n\tilde{B}B^{-1}y^{[n-1]} + \left(I - WD_n\tilde{B}B^{-1}\right)ev^T\hat{y}^{[n-2]}. \tag{75}$$

This follows immediately from direct substitution of Equation 64 into Equation 29 to obtain

$$\hat{y}^{[n-1]} = h_{n-1}WD(\delta_n)\tilde{B}\left(B^{-1}y^{[n-1]} - B^{-1}ev^T\hat{y}^{[n-2]}\right) + WD(\delta_n)\hat{e}_1v^T\hat{y}^{[n-2]},$$

where a standard form for $\tilde{V}$ is assumed and I represents the identity matrix of appropriate dimensionality.

FIRST APPROXIMATELY SAME AS LAST (FASAL) MODE

Runge-Kutta methods for which the first stage of the new step equals the last stage of the previous step have been know for some time (see, for example, Reference 18), and the property is called First Same As Last (FSAL). Its primary use is for error estimation, as for example with Dormand-Prince pairs (Reference 26), where an embedded method of higher order is created by adding an extra stage with A coefficients identical to the b coefficients of the first method. The higher order result is actually used to continue the integration in this case but seven stages are utilized for a 5th order method. No additional function evaluations are needed for successful steps and the work is the same as though only 6 stages were used because the final stage is identical to the first stage of the following step (FSAL) which would have to be evaluated anyway. Because of the high stage order, all types of DIMSIMs for which $c_1 = 0$ and $c_p = 1$ and the local stage order condition (Equation 41) is true have a First Approximately Same As Last (FASAL) property, and this can be used to save at least one internal stage evaluation on every step after the first. As

discussed above in conjunction with error estimation, theoretical work concerning the effect of global error on local stage error remains to be done, but experimental evidence for the validity of this assumption is abundant.

The local stage order condition requires that

$$Y_p^{[n-1]} = y(t_{n-2} + h_{n-1}; t_{n-2}, y_{n-2}) + O\left(h_{n-1}^{p+1}\right),$$

where here y denotes the local solution of the ODE through the indicated initial point, while

$$Y_1^{[n]} = y(t_{n-1}; t_{n-1}, y_{n-1}) + O\left(h_n^{p+1}\right),$$

where y again is a local solution, and $h_n = h_{n-1}$ and $t_{n-1} = t_{n-2} + h_{n-1}$. Note that the order condition implies that

$$y(t_{n-1}; t_{n-1}, y_{n-1}) = y(t_{n-2} + h_{n-1}; t_{n-2}, y_{n-2}) + O\left(h_{n-1}^{p+1}\right),$$

and thus we have, with $h = h_n = h_{n-1}$,

$$Y_1^{[n]} = Y_p^{[n-1]} + O\left(h^{p+1}\right).$$

Since what is used is the always the derivative multiplied times the step size, and

$$hf\left(t_{n-1}, Y_1^{[n]} + O\left(h^{p+1}\right)\right) = hf\left(t_{n-1}, Y_1^{[n]}\right) + O\left(h^{p+2}\right),$$

it is possible to use $F\left(Y_p^{[n-1]}\right)$ in place of carrying out the function evaluation for explicit methods, or even the nonlinear equation solving required in implicit methods, for $F\left(Y_1^{[n]}\right)$.

It should be noted that higher order terms ($O(h^{p+2})$) will be changed. In some cases they will be fortuitously decreased, but in others they will be increased. The smaller the error constant the greater the impact that these terms will have. But in general the error estimation, rescaling, and interpolation for the original DIMSIM should be essentially unchanged.

But the most significant impact will be on stability. It should be noted that when this implementation is used, the pth internal stage from the previous step is carried over into the next step along with the external stage vector, and thus in a sense this becomes a "two-step DIMSIM." The modified method may be written in standard General Linear Method (GLM) notation by producing $\hat{A}$, $\hat{B}$, $\hat{U}$, and $\hat{V}$ matrices from the A, B, U(= I), and V from the original DIMSIM, and by enlarging the external stage vector length by 1 to r = p + 1. This becomes a GLM with p = q = s, r = p + 1. The first component of the external stage vector becomes the pth internal stage that is carried along, while the others are unchanged. We set $\hat{A} = A$ except for the first row which becomes 0; this is unchanged for explicit methods. $\hat{U}$ is an augmentation of I produced by adding a 0 second column, and hence is px(p + 1). $\hat{B}$ is produced by adding to B the last row of A as a new first row and is (p + 1)xp. Finally, $\hat{V}$ is produced by adding to V a new first row and first column, with 1 for the last element of the first row and 0s elsewhere for these new elements. A tableau then appears as

$$\begin{bmatrix} \hat{A} & \hat{U} \\ \hat{B} & \hat{V} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ A_2 & 0 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & 0 \\ A_p & 0 & 0 & \cdots & 0 & 1 \\ A_p & & 0 & \cdots & 1 & \\ B & & & 0 \quad V & & \end{bmatrix},$$

where A_k is used to denote the kth row of matrix A.

The original V becomes the lower right submatrix and the new matrix is (p + 1) x (p + 1). Then we may write for a single ODE,

$$Y^{[n]} = h\hat{A}F\left(Y^{[n]}\right) + \hat{U}y^{[n-1]},$$
$$y^{[n]} = h\hat{B}F\left(Y^{[n]}\right) + \hat{V}y^{[n-1]}. \tag{76}$$

(It should be noted here that the external stages are redefined, although notation has not been changed, and that actual implementation will not involve separate evaluations or function evaluations for the repeated stages.) Then the new stability matrix becomes, through application to the test problem y' = λy and setting hλ = z,

$$M = z\hat{B}\left(I - z\hat{A}\right)^{-1}\hat{U} + \hat{V}. \tag{77}$$

This is now (p + 1) x (p + 1) and in general does not produce the same stability region as the original DIMSIM. Thus if FASAL implementation is desired, a method should be sought that optimizes the FASAL stability region rather than that of the DIMSIM, and in general it should not be expected that FASAL implementation for a given DIMSIM should produce a favorable region. This may be used in various ways, including development of new methods especially for FASAL implementation, or in variable implementation where FASAL is used until it runs into stability problems. Typical stability region plots will be shown below as the use of specific methods are discussed.

DEVELOPING EXPLICIT DIMSIM ODE SOLVERS

Explicit solvers have been developed based on Type 1 DIMSIMs derived by Butcher (Reference 1) and by Butcher and Jackiewicz (References 6, 7, and 8). These methods are designed to have the same desirable stability regions as Runge-Kutta methods but break Runge-Kutta order barriers at orders above 4 and stage order barriers, thus reducing the number of function evaluations required per integration step. These DIMSIMs are designed to have a number of stages (the number of function evaluations) equal to the order of the method, while explicit Runge-Kutta methods require one extra stage for orders 5 and 6, two extra for order 7, and at least three extra for orders 8 and higher. In fact, order 10 is the highest-order explicitly constructed explicit method found so far, and the fewest stages required of any 10th order method is 17 according to the current (1993) revision of Hairer, Norsett, and Wanner (Reference 18). Thus the recent announcement of the discovery of an 8th order Type 1 DIMSIM using 8 stages and with good stability properties by Butcher, Jackiewicz and Mittelmann (Reference 6) is of great significance.

The development of a solver involves derivation and testing of additional implementation parameters to provide for rescaling at step-size changes, error estimation, interpolation, and starting procedure. A solver also requires software design and implementation. In this case the final codes were written in FORTRAN 77.

IMPLEMENTION PARAMETERS FOR A SECOND ORDER EXPLICIT DIMSIM

Although second order does not provide sufficient accuracy to be broadly useful, the simplicity of the small number of parameters enables development and convenient illustration of techniques applicable to higher orders.

Butcher's second order Type 1 method in Equation 15 was utilized to develop a DIMSIM variable step-size solver. For convenience the tableau is reproduced here.

$$\begin{bmatrix} A & U \\ B & V \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ \frac{5}{4} & \frac{1}{4} & \frac{1}{2} & \frac{1}{2} \\ \frac{3}{4} & -\frac{1}{4} & \frac{1}{2} & \frac{1}{2} \end{bmatrix},$$

and c = $[0,1]^T$. Butcher (Reference 1) also found matrices W, $\tilde{B}$, and $\tilde{V}$ (see above in the Introduction to DIMSIMs section for the relevant definitions). These are

$$W = \begin{bmatrix} 1 & 0 & 0 \\ 1 & -1 & \frac{1}{2} \end{bmatrix},$$

$$\tilde{B} = \begin{bmatrix} \frac{5}{4} & \frac{1}{4} \\ 0 & 1 \\ -1 & 1 \end{bmatrix}, \tag{78}$$

$$\tilde{V} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

The latter two were determined using a free parameter g to obtain desirable step-size changing zero stability, determined from the eigenvalues of the matrix

$$WD\tilde{V} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} - \delta g + \delta^2 g & \frac{1}{2} + \delta g - \delta^2 g \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} = V,$$

for g = 0, which has eigenvalues 0 and 1, independent of δ. Note that standard form is used for $\tilde{V}$ in this case. The stability polynomial $M(z) = 1 + z + z^2/2$, is the same as for a two-stage, second-order explicit Runge-Kutta method, and thus this method has the same well-known stability region, including the interval [-2, 0] along the real axis.

Butcher (Reference 1) also derived estimates for the local truncation error (after the first step) as

$$lte \cong \frac{2\delta}{1+\delta}\left(\tfrac{1}{6}hf(Y_2) - \tfrac{1}{2}hf(Y_1) + \tfrac{1}{3}\left(y_1^{[n-1]} - y_2^{[n-1]}\right)\right), \tag{79}$$

which reduces for fixed step ($\delta = 1$) to

$$lte \cong \tfrac{1}{6}hf(Y_2) - \tfrac{1}{2}hf(Y_1) + \tfrac{1}{3}\left(y_1^{[n-1]} - y_2^{[n-1]}\right).$$

For the first step error estimate we apply Theorem 8, Equation 53 and calculate

$$\frac{1}{(p+1)!} - \sum_{j=1}^{p} \tilde{B}_{1j} \frac{c_j^p}{p!} = \tfrac{1}{24}.$$

The simultaneous equations to satisfy from Equation 54 are

$$\begin{aligned}
&\gamma_1 + \gamma_2 = 0 \\
&\beta_1 + \beta_2 - \gamma_2 = 0 \\
&\beta_2 + \tfrac{1}{2}\gamma_2 = 0 \\
&\tfrac{1}{2}\beta_2 = 1
\end{aligned}$$

Then $\beta^T = [-6,2], \gamma^T = [4,-4],$ and the first step error estimate becomes

$$lte \cong \tfrac{1}{12}hf(Y_2) - \tfrac{1}{4}hf(Y_1) + \tfrac{1}{6}\left(y_1^{[0]} - y_2^{[0]}\right). \tag{80}$$

This is exactly one-half of the fixed-step error estimate used for subsequent steps.

The final computation involves the derivation of an interpolant using Theorem 5. We need to find the 12 coefficients of

$$\beta_0^T(\theta) = \begin{bmatrix} \beta_{010} + \beta_{011}\theta + \beta_{012}\theta^2 \\ \beta_{020} + B_{021}\theta + \beta_{022}\theta^2 \end{bmatrix}$$

and

$$\gamma_0^T(\theta) = \begin{bmatrix} \gamma_{010} + \gamma_{011}\theta + \gamma_{012}\theta^2 \\ \gamma_{020} + \gamma_{021}\theta + \gamma_{022}\theta^2 \end{bmatrix},$$

where $\theta \in [0,1]$ is the interpolation parameter. We find immediately from the continuity condition $\beta_0(0) = 0$ that we must have $\beta_{010} = \beta_{020} = 0$. The other continuity conditions II(3.3), when combined with the compatibility conditions (see Implementing DIMSIMs section), yield the equations

$$\gamma_0(0)B = \left[\tfrac{5}{4}\gamma_{010} + \tfrac{3}{4}\gamma_{020} \quad \tfrac{1}{4}\gamma_{010} - \tfrac{1}{4}\gamma_{020}\right] = \beta_0(1) = \left[\tfrac{5}{4} \quad \tfrac{1}{4}\right],$$

which yields $\gamma_{010}=1$, $\gamma_{020}=0$, and

$$\gamma_0(0)V = \left[\tfrac{1}{2}(\gamma_{010} + \gamma_{020}) \quad \tfrac{1}{2}(\gamma_{010} + \gamma_{020})\right] = \gamma_0(1) = \left[\tfrac{1}{2} \quad \tfrac{1}{2}\right],$$

which is consistent with the previous result but yields no additional information. The basic compatibility conditions then tell us that

$$\begin{bmatrix} \frac{5}{4} \\ \frac{1}{4} \end{bmatrix} = \begin{bmatrix} \beta_{011} + \beta_{012} \\ \beta_{021} + \beta_{022} \end{bmatrix},$$

$$\begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix} = \begin{bmatrix} \gamma_{010} + \gamma_{011} + \gamma_{012} \\ \gamma_{020} + \gamma_{021} + \gamma_{022} \end{bmatrix},$$

or, using the values just calculated, the second vector equation becomes

$$\begin{bmatrix} -\frac{1}{2} \\ \frac{1}{2} \end{bmatrix} = \begin{bmatrix} \gamma_{011} + \gamma_{012} \\ \gamma_{021} + \gamma_{022} \end{bmatrix}.$$

The order condition now becomes

$$z\left[\beta_{011}\theta + \beta_{012}\theta^2 \quad \beta_{012}\theta + \beta_{022}\theta^2\right]\begin{bmatrix} 1 \\ e^z \end{bmatrix} + \left[1 + \gamma_{011}\theta + \gamma_{02}\theta^2 \quad \gamma_{021}\theta + \gamma_{022}\theta^2\right]\begin{bmatrix} 1 \\ 1 - z + \frac{1}{2}z^2 \end{bmatrix}$$
$$= e^{\theta z} + O\left(z^3\right).$$

We expand the exponentials about z = 0 and equate coefficients of powers of z. This gives us the following three equations:

$$O(1): \; 1+\gamma_{011}\theta+\gamma_{012}\theta^2+\gamma_{021}\theta+\gamma_{022}\theta^2=1$$

$$O(z): \; \beta_{011}\theta+\beta_{012}\theta^2+\beta_{021}\theta+\beta_{022}\theta^2-\gamma_{021}\theta-\gamma_{022}\theta^2=\theta$$

$$O\left(z^2\right): \; \beta_{021}\theta+\beta_{022}\theta^2+\tfrac{1}{2}\gamma_{021}\theta+\tfrac{1}{2}\theta^2=\frac{\theta^2}{2}.$$

This gives us 7 equations for 8 variables, but we clearly must seek to eliminate the parameter dependence on θ. We use the compatibility conditions to eliminate γ_{011}, γ_{021}, β_{011}, and β_{021}. Our O(1) condition then reduces to

$$-(1-\theta)\gamma_{012}-(1-\theta)\gamma_{022}=0,$$

which is used to eliminate γ_{022} when the condition $\theta \neq 1$ is considered. The O(z) condition then yields the equation

$$1-\theta-2\beta_{022}(1-\theta)-\gamma_{022}(1-\theta)=0.$$

This similarly enables elimnation of γ_{022}. Finally, the $O(z^2)$ condition is applied. It reduces to

$$1-\theta-\beta_{012}(1-\theta)-3\beta_{022}(1-\theta)=0.$$

We then can eliminate β_{012}, leaving us with the following one-parameter family of coefficients:

$$\gamma_{010} = 1,$$

$$\gamma_{011} = \tfrac{1}{2} - 2\beta_{022},$$

$$\gamma_{012} = -1 + 2\beta_{02},$$

$$\gamma_{020} = 0,$$

$$\gamma_{021} = -\tfrac{1}{2} + 2\beta_{022},$$

$$\gamma_{022} = 1 - 2\beta_{022},$$

$$\beta_{010} = 0,$$

$$\beta_{011} = \tfrac{1}{4} + 3\beta_{022},$$

$$\beta_{012} = 1 - 3\beta_{022},$$

$$\beta_{020} = 0,$$

$$B_{021} = \tfrac{1}{4} - \beta_{022},$$

$$\beta_{022} = \beta_{022}.$$

We let $\beta_{022} = 0$ to obtain the Butcher-Jackiewicz-type interpolant

$$\begin{aligned}\eta(x_{n-1} + \theta h) = h\left(\tfrac{1}{4}\theta + \theta^2\right) f\left(Y_1^{[n]}\right) + h\frac{\theta}{4} f\left(Y_2^{[n]}\right) \\ + \left(1 + \frac{\theta}{2} - \theta^2\right)\tilde{y}_1^{[n-1]} + \left(-\frac{\theta}{2} + \theta^2\right)\tilde{y}_2^{[n-1]}.\end{aligned} \tag{81}$$

Some possible starting procedures were derived above in the chapter Implementing DIMSIMs as an illustration of techniques generally applicable to DIMSIMs. For our

explicit second order solver we use an order 3 explicit Runge-Kutta solver to obtain an approximation y_1 to $y(t_0+h_0)$ accurate to $O(h_0^4)$. Then we have:

$$y_1 = y_0 + h_0 y_0' + \frac{h_0^2}{2} y_0'' + \frac{h_0^3}{6} y_0''' + O\left(h_0^4\right),$$

$$y_1' = y_0' + h_0 y_0'' + \frac{h_0^2}{2} y_0''' + O\left(h_0^3\right).$$

Here $y_1' = f\left(t_0 + h_0, y_1\right)$. Eliminating variables we solve these two simultaneous equations and obtain

$$y_0'' = \frac{2}{h_0^2}\left(3\left(y_1 - y_0\right) - 2y_0' - y_1'\right) + O\left(h_0^2\right),$$

$$y_0''' = \frac{6}{h_0^3}\left(2\left(y_0 - y_1\right) + y_0' + y_1'\right) + O\left(h_0\right).$$

Note that the third derivative is not needed for the Nordsieck vector but is useful in selecting initial step size. Then the Nordsieck vector at t_0 is given by

$$\tilde{y}(t_0) = \begin{bmatrix} y(t_0) \\ h_0 y'(t_0) \\ h_0^2 y''(t_0) \end{bmatrix} = \begin{bmatrix} y_0 \\ h_0 f(t_0, y_0) \\ 6\left(y_1 - y_0\right) - 4y_0' - 2y_1' \end{bmatrix} + O\left(h_0^4\right).$$

Note that this depends on the step size. Once a correct step size h_1 is chosen, this vector is rescaled using the matrix

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \delta & 0 \\ 0 & 0 & \delta^2 \end{bmatrix},$$

where δ is the ratio $\dfrac{h_1}{h_0}$. Multiplication by W then yields the appropriate starting vector. The error in the first step is given by Theorem 8 to be

$$lte_1 = \left(\frac{1}{3!} - \sum_{j=1}^{2} B_{1j} \frac{c_j^2}{2!} \right) y^{(3)}(t_0) h_1^3 + O\left(h_1^4\right) = \tfrac{1}{4}\delta^3 \left(2\left(y_0 - y_1\right) + h_0\left(y_0' + y_1'\right)\right) + O\left(h_1^4\right).$$

If we set this error to half the tolerance τ and use a norm to include the possibility of systems of equations, we then may write a conservative but hopefully accurate choice for initial step size to be δh_0, where

$$\delta = \left(\frac{2\tau}{\left\| 2\left(y_0 - y_1\right) + h_0\left(y_0' + y_1'\right) \right\|} \right)^{\frac{1}{3}}. \tag{82}$$

TESTING IMPLEMENTATION PARAMETERS FOR SECOND ORDER

A test equation first proposed by Prothero and Robinson (Reference 27) was extensively utilized:

$$y' = \lambda\left(y - p(t)\right) + p'(t), \;\; y\left(t_0\right) = y_0. \tag{83}$$

The exact solution is

$$y(t) = \left(y_0 - p\left(t_0\right)\right) \exp\left(\lambda\left(t - t_0\right)\right) + p(t). \tag{84}$$

It is interesting to note that for $y_0 = p\left(t_0\right)$ the solution is simply p(t). The sine function was used for p(t) and the interval of interest is $t \in [0, 20]$. Typical values for λ were -2 and -20.

A quantity r is determined for each step, where r is defined as

$$r \equiv \left| \left\| \frac{err}{est} \right\| - 1 \right|. \tag{85}$$

Here err represents the local error as described in Equation 40, that is, the solution of the initial value problem in Equation 1 beginning at a DIMSIM solution point (t_{n-1}, y_{n-1}) with exact local solution at t_n given by $y(t_n; t_{n-1}, y_{n-1})$, and with calculated solution at t_n of y_n. Thus we test how well the estimate approximates the error using Equation 40,

$$err_n = y_n - y(t_n; t_{n-1}, y_{n-1}).$$

It should be noted that it is not appropriate to restart the solution in seeking to obtain the expression for *err*. A DIMSIM changes character after the first step, and this must be preserved in the testing process. In particular the external stage vector is not recalculated using a starting procedure, otherwise the separate error estimate for an intial step would be required. Testing will only then truly reveal the behavior of the error estimator in its use within a solver.

The results of a test may be indicated both graphically and using a number of statistics. The quantity *est* is used to denote the result of the use of the error estimate provided for the method. A graph showing both *err* and *est* together is sometimes very revealing, but it is also vital to look at the largest error and the solution range. And a table showing the percentages of error estimates yielding r less than 1%, 5%, 10%, 25%, 50%, and 100% has proven to be of great value. Of course it is expected that test results will vary greatly with the tolerance used and also with the problem solved and even with the problem parameters. Thus any testing of implementation parameters must be considered preliminary to the much more extensive testing required of a full solver.

Two tests were provided for the quality of error estimates. Test 2 was carried out as part of the testing of the complete solver on a number of different problems and is described with results in the Developing Explicit DIMSIM ODE Solvers section. Test 1 was first described by Butcher and Jackiewicz (Reference 3). A scheme was devised for testing the effects of rapid step changes on error estimation accuracy. An initial step size h_0 is chosen, along with a ratio ρ. The cyclic pattern of step sizes h_0, ρh_0, $\rho^2 h_0$, ρh_0, h_0, h_0/ρ, h_0/ρ^2, h_0/ρ, h_0, ..., is used until the end of the interval of interest is reached. This should be considered to be a very stringent test, especially for higher values of ρ, and serves as an excellent preliminary check. The behavior even with rapid step-size changing is quite good, as shown in Tables 1 and 2, and Figure 1.

TABLE 1. Error Estimate Test for Second Order, Part 1.

ρ	% r < 0.01	%r < 0.05	%r < 0.10	%r < 0. 25	%r < 0.50	%r < 1.0
1.25	0.66	9.27	21.19	49.67	95.36	99.34
1.50	3.29	11.84	23.68	49.34	91.45	99.34
1.75	2.63	6.58	13.16	34.87	78.29	99.34
2.00	0.65	3.92	9.80	25.49	67.32	99.35

TABLE 2. Error Estimate Test for Second Order, Part 2.

ρ	rmin	rmax	t_f	max err	max err/Δt_f
1.25	0.2811	3.2740	20.0402	0.0011	0.0610
1.50	0.0031	6.6206	20.0963	0.0010	0.0584
1.75	0.1719	3.8194	20.0335	0.0010	0.0597
2.00	0.0402	22.0300	20.1128	0.0010	0.0559

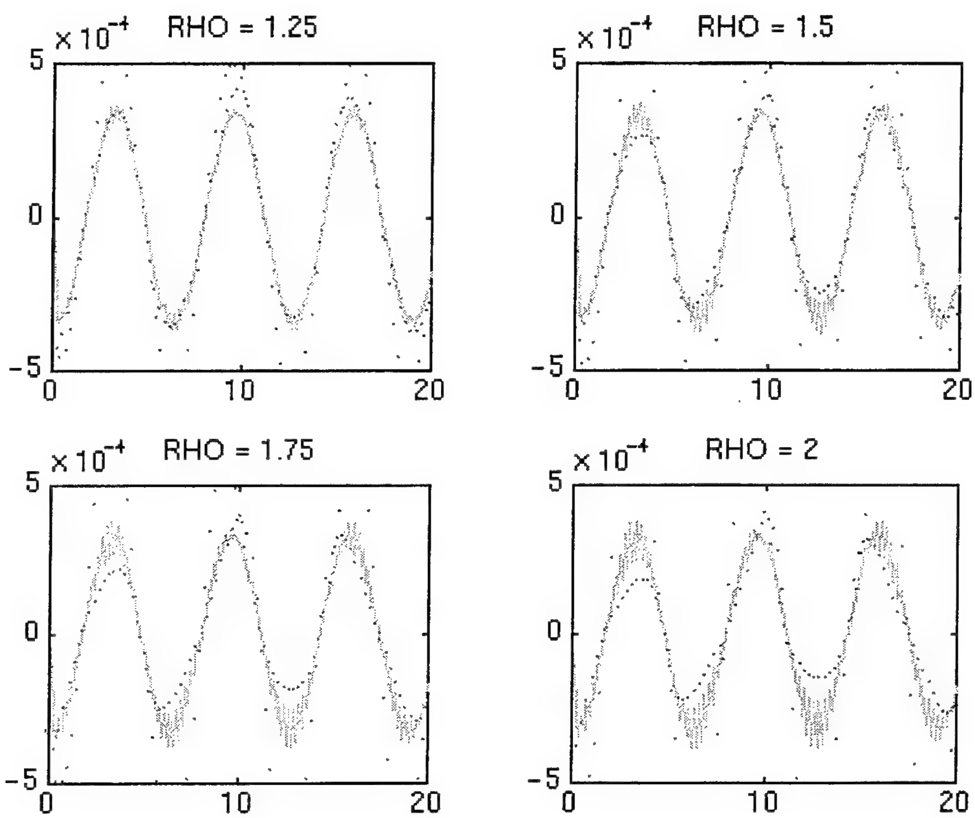


FIGURE 1. Graph of Error Estimate Test for Second Order. Estimate is dotted line, local error is solid line.

A related test was devised for interpolants. The interpolant was used to calculate the solution at 10 points evenly spaced across a step. The error at the end of the step was used to scale the error at the interpolated points. Thus a ratio no greater than 1 in absolute value would indicate that the error at the interpolated point is no greater than the error at the end point. Any of the error estimator tests described above might be used to generate the integration points, but the rapid step-size changing test seemed most directly useful.

Figure 2 shows the behavior of the Butcher-Jackiewicz-type interpolant (Equation 81) error relative to the behavior of the solver at the grid points. It performs very well with

only a few of the thousands of interpolated points having an error greater than the error at the grid points. In fact, much of the time the error at the interpolated points in the middle of the interval is less than the end point error. Certain grid points seem to lie close to a zero of the error function, having associated errors as much as 2 to 4 times less than those at other points, and the interpolated values in the middle of those intervals have significantly less accuracy than the grid points.

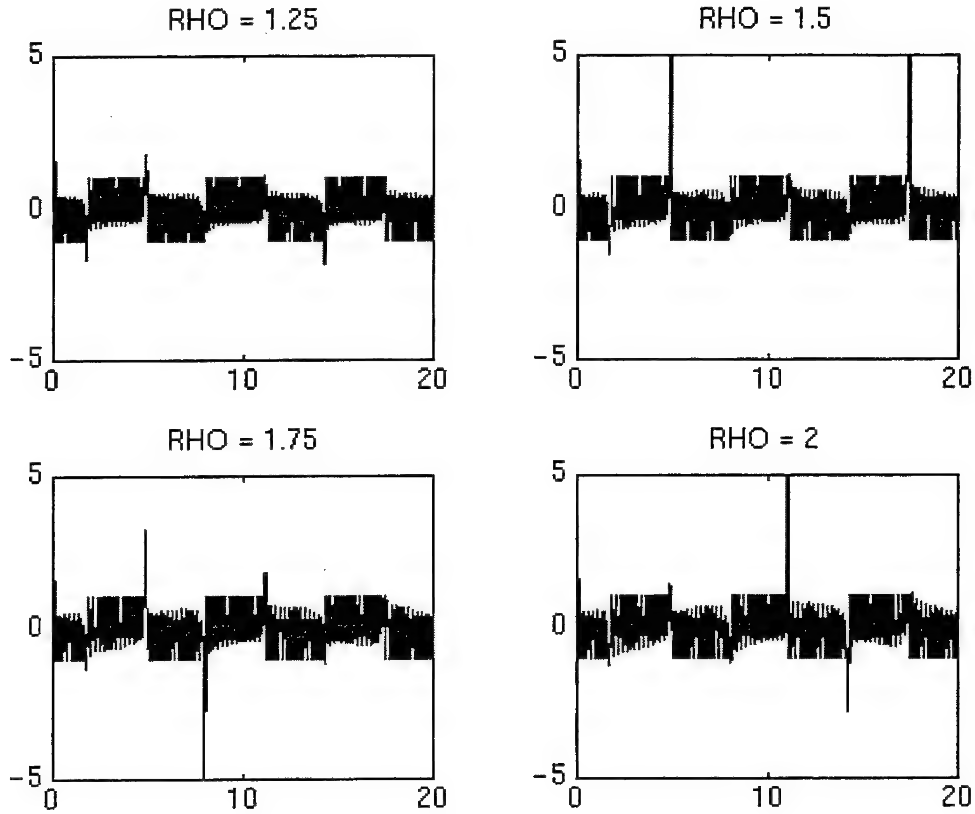


FIGURE 2. Test Results for Order 2 Butcher-Jackiewicz-type Interpolant.

It is interesting to look at the percentage of relative errors falling in various ranges. For ~1500 steps the factors k by which the interpolation errors were greater than the grid point error were distributed as shown in Table 3.

TABLE 3. Test Results for Order 2 Butcher-Jackiewicz-Type Interpolant.

ρ	k > 10	10 > k > 5	5 > k > 2	2 > k > 1	k <= 1
1.25	0.0%	0.0%	0.0%	1.8%	98.2%
1.50	0.6%	0.4%	0.1%	0.9%	98.0%
1.75	0.0%	0.1%	0.8%	0.4%	97.7%
2.00	0.3%	0.2%	0.4%	1.5%	97.6%

The Nordsieck interpolant was also tested using the same test problem. Figure 3 shows that this interpolant has errors that are fairly comparable to the grid point errors. The simple second order Nordsieck vector was used without the third order continuity correction term.

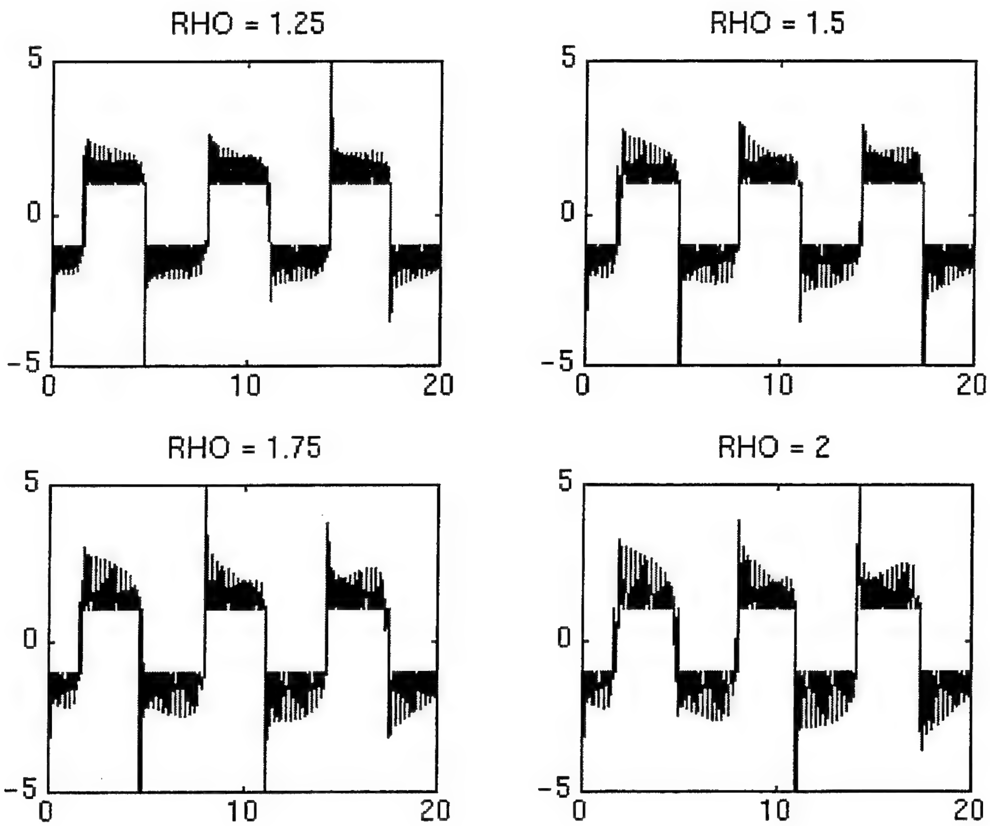


FIGURE 3. Order 2 Nordsieck Interpolant Test.

For ~1500 steps the factors k by which the interpolation errors were greater than the grid point error were distributed as shown in Table 4.

TABLE 4. Order 2 Nordsieck Interpolant Test.

ρ	k > 10	10 > k > 5	5 > k > 2	2 > k > 1	k <= 1
1.25	0.0%	0.1%	10.2%	78.3%	11.3%
1.50	0.9%	0.1%	13.1%	74.7%	11.2%
1.75	0.1%	0.5%	15.5%	72.6%	11.2%
2.00	0.4%	0.4%	17.1%	70.6%	11.6%

It is evident that for this method, the interpolant of Butcher-Jackiewicz type (Equation 81) provides a better representation of the solution than the simple 2nd order Nordsieck interpolant.

The continuous version was then tried with the graphical results shown in Figure 4.

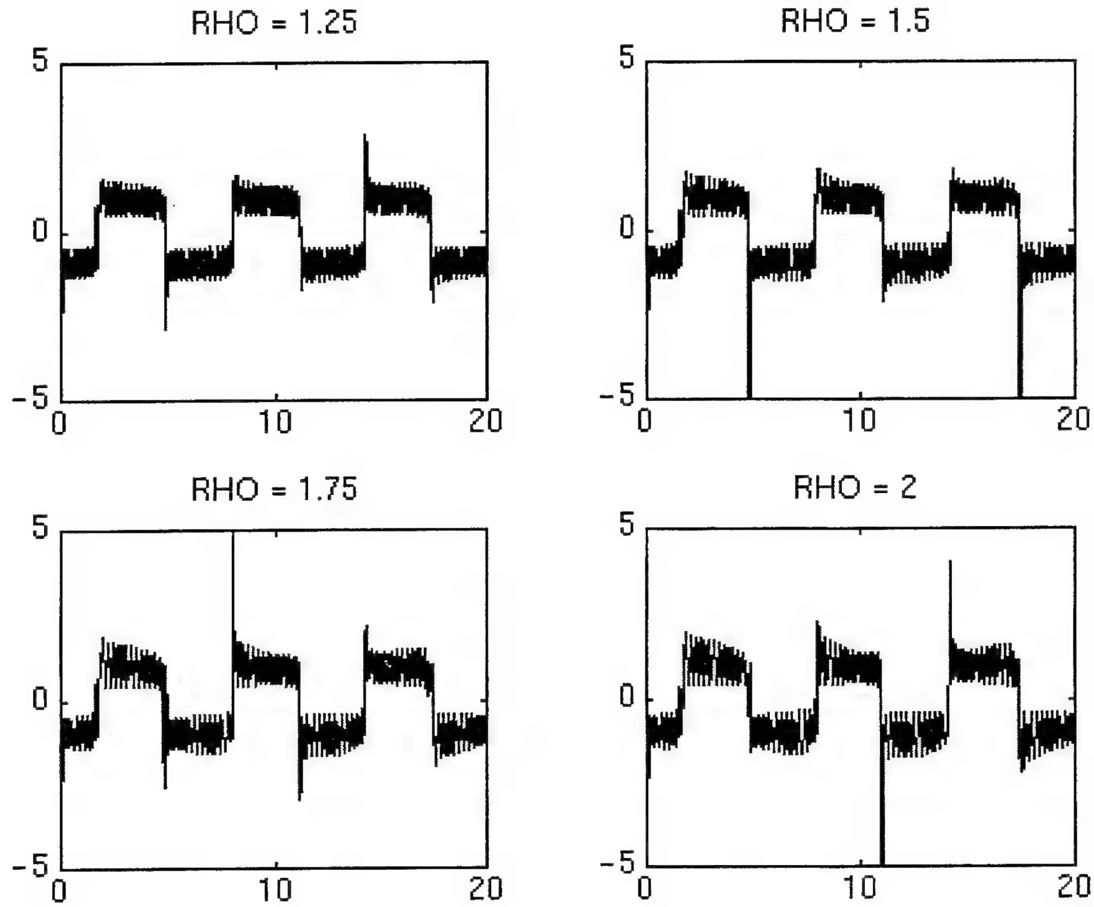


FIGURE 4. Order 2 Continuous Nordsieck Interpolant Test.

The statistical breakdown obtained by continuous Nordsieck interpolant test is shown in Table 5.

TABLE 5. Order 2 Continuous Nordsieck Interpolant Test.

ρ	$k > 10$	$10 > k > 5$	$5 > k > 2$	$2 > k > 1$	$k <= 1$
1.25	0.0%	0.0%	1.2%	68.1%	30.7%
1.50	0.6%	0.4%	0.5%	67.8%	30.7%
1.75	0.0%	0.2%	1.5%	67.1%	31.2%
2.00	0.3%	0.2%	1.2%	66.7%	31.6%

This interpolant is less accurate here than the Butcher-Jackiewicz-type interpolant (Equation 81), but provides substantially better accuracy than the 2nd order Nordsieck interpolant and could be readily used if a better alternative did not exist for this case.

The testing of the starting method concerns the accuracy of the starting vector, the accuracy of the first step error estimate, and the appropriateness of the choice for initial step size. The Prothero-Robinson test problem is used here with $\lambda = -2$, with a starting point at t_0 on the exact solution trajectory through (0, 1).

The following test reveals the accuracy of the error estimate and of the initial step size selection algorithm. The starting point at $t_0 = 1$ is used throughout to avoid the special case of a 0 third derivative at t_0. It is evident that accuracy improves until round-off error becomes significant. The fact that the error term is $O(h^3)$ is evident as a tightening of the tolerance by a factor of 1000 produces changes of a factor of 10 in h.

TABLE 6. Order 2 Starting Procedure Test 1.

Tol	h (Nord)	Loc err	Err est	h (H-N-W)	Loc err	Err est
10^{-3}	$1.95\text{x}10^{-1}$	$-3.21\text{x}10^{-4}$	$-5.04\text{x}10^{-4}$	$3.09\text{x}10^{-2}$	$-1.87\text{x}10^{-6}$	$-1.99\text{x}10^{-6}$
10^{-6}	$1.95\text{x}10^{-2}$	$-4.81\text{x}10^{-7}$	$-5.01\text{x}10^{-7}$	$3.09\text{x}10^{-3}$	$-1.97\text{x}10^{-9}$	$-1.99\text{x}10^{-9}$
10^{-9}	$1.95\text{x}10^{-3}$	$-4.98\text{x}10^{-10}$	$-5.00\text{x}10^{-10}$	$3.09\text{x}10^{-4}$	$-2.02\text{x}10^{-12}$	$-2.00\text{x}10^{-12}$
10^{-12}	$1.95\text{x}10^{-4}$	$-5.00\text{x}10^{-13}$	$-5.00\text{x}10^{-13}$	$3.09\text{x}10^{-5}$	$-1.89\text{x}10^{-15}$	$-1.97\text{x}10^{-15}$
10^{-15}	$1.95\text{x}10^{-5}$	$-5.55\text{x}10^{-16}$	$-5.11\text{x}10^{-16}$	$3.09\text{x}10^{-6}$	$-1.11\text{x}10^{-16}$	$-3.75\text{x}10^{-18}$

Another revealing test uses starting points at 0 through 1, evenly spaced, and with a fixed tolerance of 10^{-9}. We note in Table 7 that the use of the Nordsieck-vector-based error estimate for the first step provides a much more efficient start than the method described above in the previous chapter as outlined in Hairer, Norsett and Wanner (Reference 18).

TABLE 7. Order 2 Starting Procedure Test 2.

t_0	h (Nord)	Loc err	Err est	h (H-N-W)	Loc err	Err est
0	1.10×10^{-03}	-4.99×10^{-10}	-5.00×10^{-10}	1.36×10^{-04}	-9.38×10^{-13}	-9.39×10^{-13}
0.2	1.24×10^{-03}	-4.99×10^{-10}	-5.00×10^{-10}	1.59×10^{-04}	-1.07×10^{-12}	-1.07×10^{-12}
0.4	1.39×10^{-03}	-4.99×10^{-10}	-5.00×10^{-10}	2.11×10^{-04}	-1.77×10^{-12}	-1.78×10^{-12}
0.6	1.55×10^{-03}	-4.98×10^{-10}	-5.00×10^{-10}	2.55×10^{-04}	-2.24×10^{-12}	-2.24×10^{-12}
0.8	1.73×10^{-03}	-4.98×10^{-10}	-5.00×10^{-10}	3.24×10^{-04}	-3.29×10^{-12}	-3.29×10^{-12}
1	1.95×10^{-3}	-4.98×10^{-10}	-5.00×10^{-10}	3.09×10^{-4}	-2.02×10^{-12}	-2.00×10^{-12}

The final test concerns the calculation of the starting vector. Here we are concerned that the smallest possible safe value for h_0 be utilized to produce the maximum accuracy since no DIMSIM integration step will actually be taken. We estimate that the derivatives will all be approximately at least the same size as $g(t_0,y_0)=\min\left(\|y_0\|,\|f(t_0,y_0)\|\right)$, but taking the maximum if the minimum is smaller by a factor of eps, the machine epsilon. We note that quantities $h_0^k y^{(k)}(t_0)$ will be calculated in terms of linear combinations of functional values and derivatives multiplied by h_0, with the most difficult term to calculate corresponding to k=p+1, used for the initial step-size selection. We would like to have at least three significant digits for this derivative, and we expect to have more correct digits for the other terms. Thus we would like to have $g(t_0,y_0)h_0^{p+1}=10^5 eps$ (10^3 instead of 10^5 worked well for order 2 but not for order 5, this is a bit more conservative but more generally useful). This yields an optimal choice of $h_0=10^{\frac{5}{p+1}}\left(\frac{eps}{g(t_0,y_0)}\right)^{\frac{1}{p+1}}$. In this case of course, p = 2. We test this by calculating the relative errors in the second and third derivatives at 5 uniformly spaced starting points between 0 and 1, and also looking at the maximum norm of the error in the starting external stage vector. Note that for this test problem the increment was quite appropriate (see Table 8).

TABLE 8. Order 2 Starting Procedure Test 3.

t_0	h_0	$y''(t_0)$	Rel err	$y'''(t_0)$	Rel err	$\left\|\Delta y^{[0]}\right\|_\infty$
0	2.81×10^{-4}	4.00	1.06×10^{-7}	-9.00	5.02×10^{-4}	1.69×10^{-14}
0.2	3.95×10^{-4}	2.48	2.12×10^{-7}	-6.34	6.47×10^{-4}	4.11×10^{-14}
0.4	9.97×10^{-4}	1.41	1.44×10^{-6}	-4.51	1.46×10^{-3}	1.01×10^{-12}
0.6	4.64×10^{-4}	6.40×10^{-1}	3.64×10^{-7}	-3.23	5.68×10^{-4}	2.51×10^{-14}
0.8	4.23×10^{-4}	9.02×10^{-2}	8.32×10^{-7}	-2.31	3.94×10^{-4}	6.77×10^{-15}
1	4.35×10^{-4}	-3.00×10^{-1}	2.78×10^{-8}	-1.62	2.45×10^{-4}	7.77×10^{-16}

IMPLEMENTATION PARAMETERS FOR A FIFTH ORDER EXPLICIT DIMSIM

A fifth order type 1 DIMSIM derived by Butcher and Jackiewicz (Reference 8) was adapted for implementation as an ODE solver. The method was derived using numerical techniques for solving the large system of nonlinear equation and, to ensure favorable stability, adding a requirement that the form of the stability polynomial reduce to the standard 5th order Runge-Kutta stability polynomial. Here again, p = q = r = s = 5, U = I. The method coefficients are as follows:

$$c = \begin{bmatrix} 0 & \frac{1}{4} & \frac{1}{2} & \frac{3}{4} & 1 \end{bmatrix}^T,$$

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1.1765281703106688 & 0 & 0 & 0 & 0 \\ 1.9805793463233191 & .4017181027378085 & 0 & 0 & 0 \\ 3.0532835108392395 & .7349961462028882 & .2672357626475791 & 0 & 0 \\ 1.4193325269467698 & 2.6534897473125331 & -2.2778532945468265 & 1.1978905088172778 & 0 \end{bmatrix}$$

V is given by a matrix with 5 identical rows, where each row is v^T, and

$$v = \begin{bmatrix} -.2406956155386215 \\ 1.2604945758471451 \\ -2.4812693924523267 \\ 1.9199070083032958 \\ 0.5415634238405073 \end{bmatrix}.$$

$$B = \begin{bmatrix} 3.1630239143645557360 & 1.974339246050540681 & -0.81042512005562813 & 0.540922018802018401 & 0.055078654196316838440 \\ 3.250176692142333514 & 1.53197813493942957 & 0.097908213277705203 & -0.422272425642426043 & -0.461380071669907517 \\ 3.508528033848969459 & 0.327374204184027625 & 2.239060519232953536 & -2.097452509375452155 & -0.936868983593822539 \\ 4.176223954923772036 & -2.618271719393119920 & 7.03900409877443037 & -5.2884360132659356 & -1.691097027371050162 \\ 3.008220785304765914 & 2.11453341958770507 & 0.3572048837825639 & -1.904253308575986040 & -0.645626555270999520 \end{bmatrix}$$

and was calculated using Theorem 2 and Mathematica. The computation of W is straightforward using the first relation of Equation 11,

$$W\begin{bmatrix} 1 \\ z \\ z^2 \\ z^3 \\ z^4 \\ z^5 \end{bmatrix} = (I - zA)e^{cz} + O\left(z^6\right).$$

This yields, equating terms of the same degree,

$$W = \begin{bmatrix} e & c - Ae & \dfrac{c^2}{2} - Ac & \dfrac{c^3}{6} - A\dfrac{c^2}{2} & \dfrac{c^4}{24} - A\dfrac{c^3}{6} & \dfrac{c^5}{120} - A\dfrac{c^4}{24} \end{bmatrix},$$

or evaluating this numerically,

$$W = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -0.926528170310669 & 0.03125 & 0.002604166666666666 & 0.0001627604166666666 & 8.138020833333310e-6 \\ 1 & -1.882297449061128 & 0.02457047431554788 & 0.00827964262277682 & 0.001558025774120291 & 0.0001950328608825182 \\ 1 & -3.305515419689707 & -0.0361169178745116 & 0.01393940010021236 & 0.005702129564105415 & 0.001161984318267553 \\ 1 & -1.992859488529754 & 0.07713632883232162 & 0.03157006827664394 & -0.002014862315115681 & -0.001959141967572072 \end{bmatrix}$$

This approach was utilized in calculating the rescaling matrices for this 5th order DIMSIM. The customary form of $\tilde{V}$ is utilized with a first row of v^T and the remaining elements 0, while

$$\bar{B} = \begin{bmatrix} 3.163023914364555 & 1.974339246050541 & -0.810425120055628 & 0.5409220188020184 & 0.05507865419631683\,1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & -\frac{16}{3} & 12 & -16 & \frac{25}{3} \\ \frac{44}{3} & -\frac{224}{3} & 152 & -\frac{416}{3} & \frac{140}{3} \\ 96 & -448 & 768 & -576 & 160 \\ 256 & -1024 & 1536 & -1024 & 256 \end{bmatrix}$$

was computed using Mathematica.

Interpolation may always be carried out using the Nordsieck vector or the continuous modification of the Nordsieck interpolant as described above in the Implementing DIMSIMs section. However, the Butcher-Jackiewicz interpolant has some desirable

features when it exists, and it was also calculated for this method. It is a continuous interpolant of both order and degree 5, while the simple Nordsieck interpolant is also of order and degree 5 but not continuous at the grid points, and the continuous Nordsieck interpolant is of degree 6 but adds the additional oscillation typical of a higher degree polynomial without increasing the order of accuracy, since the method itself is only of order 5. The interpolant sought is of the form Equation 34, which in this case is

$$\eta(t_{n-1}+\theta h_n)=h_n\beta_0(\theta)F\left(Y^{[n]}\right)+\gamma_0(\theta)\hat{y}^{[n-1]},$$

where β_0 and γ_0 are both column vectors of polynomials of degree 5. We write them in the form

$$\beta_0=\begin{bmatrix}\beta_{010}+\beta_{011}\theta+\beta_{012}\theta^2+\beta_{013}\theta^3+\beta_{014}\theta^4+\beta_{015}\theta^5\\ \beta_{020}+\beta_{021}\theta+\beta_{022}\theta^2+\beta_{023}\theta^3+\beta_{024}\theta^4+\beta_{025}\theta^5\\ \beta_{030}+\beta_{031}\theta+\beta_{032}\theta^2+\beta_{033}\theta^3+\beta_{034}\theta^4+\beta_{035}\theta^5\\ \beta_{040}+\beta_{041}\theta+\beta_{042}\theta^2+\beta_{043}\theta^3+\beta_{044}\theta^4+\beta_{045}\theta^5\\ \beta_{050}+\beta_{051}\theta+\beta_{052}\theta^2+\beta_{053}\theta^3+\beta_{054}\theta^4+\beta_{055}\theta^5\end{bmatrix}^T$$

and

$$\gamma_0=\begin{bmatrix}\gamma_{010}+\gamma_{011}\theta+\gamma_{012}\theta^2+\gamma_{013}\theta^3+\gamma_{014}\theta^4+\gamma_{015}\theta^5\\ \gamma_{020}+\gamma_{021}\theta+\gamma_{022}\theta^2+\gamma_{023}\theta^3+\gamma_{024}\theta^4+\gamma_{025}\theta^5\\ \gamma_{030}+\gamma_{031}\theta+\gamma_{032}\theta^2+\gamma_{033}\theta^3+\gamma_{034}\theta^4+\gamma_{035}\theta^5\\ \gamma_{040}+\gamma_{041}\theta+\gamma_{042}\theta^2+\gamma_{043}\theta^3+\gamma_{044}\theta^4+\gamma_{045}\theta^5\\ \gamma_{050}+\gamma_{051}\theta+\gamma_{052}\theta^2+\gamma_{053}\theta^3+\gamma_{054}\theta^4+\gamma_{055}\theta^5\end{bmatrix}^T.$$

We use the criterion Equations 35 and 36 to evaluate the unknown coefficients. That $\beta_{0j0}=0$ for all j follows immediately from the condition $\beta_0(0)=0$. From Corollary 1.ii to Theorem 5 we find that we must have $\gamma_0(0)=[1\quad 0\quad 0\quad 0\quad 0]$. This reduces the continuity conditions of Equation 36 and to the following relations:

$$\sum_{j=1}^{5}\beta_{0ij}=b_{1,i},\ \ i=1,2,\ldots,5$$

and

$$\sum_{j=0}^{5} \gamma_{0ij} = v_i, \ i = 1,2,\ldots,5.$$

There are a total of 50 unknowns remaining and 10 linear equations have already been specified. The order condition in Equation 35 provides additional linear equations. We have, again,

$$z\beta_0(\theta)e^{cz} + \gamma_0(\theta)w(z) = e^{\theta z} + O\left(z^6\right), \ \theta \in (0,1].$$

We find the following equations for terms of degree 0 through 5 in z, note that the alphas are the elements of W and that the first row of W is 0 except for a first element 1.

<u>Degree 0</u>: $\gamma_0(\theta)\alpha_0 = \gamma_0(\theta)e = 1.$

<u>Degree 1</u>: $\beta_0(\theta)e + \sum_{i=2}^{5} \gamma_{0i}(\theta)\alpha_{i1} = \theta.$

<u>Degree 2</u>: $\sum_{i=2}^{5} \beta_{0i}(\theta)c_i + \sum_{i=2}^{5} \gamma_{0i}(\theta)\alpha_{i2} = \frac{\theta^2}{2}.$

<u>Degree 3</u>: $\sum_{i=2}^{5} \beta_{0i}(\theta)\frac{c_i^2}{2} + \sum_{i=2}^{5} \gamma_{0i}(\theta)\alpha_{i3} = \frac{\theta^3}{6}.$

<u>Degree4:</u> $\sum_{i=2}^{5} \beta_{0i}(\theta)\frac{c_i^3}{6} + \sum_{i=2}^{5} \gamma_{0i}(\theta)\alpha_{i4} = \frac{\theta^4}{24}.$

<u>Degree 5</u>: $\sum_{i=2}^{5} \beta_{0i}(\theta)\frac{c_i^4}{24} + \sum_{i=2}^{5} \gamma_{0i}(\theta)\alpha_{i5} = \frac{\theta^5}{120}.$

Setting terms of the same degree in q in each of these equations to 0 provides 5 equations for each degree, for a total of 30 additional equations. This means that there are a total of 40 equations to solve with 50 unknowns, leaving 10 free parameters.

The equations were solved using Mathematica using a process of symbolic elimination of variables. The decimal representation of the method parameters resulted in the appearance of some inexact floating point numbers. When the coefficients or constant terms were 1.0e - 16 or less they were understood to be 0 for the purpose of satisfying the interpolant defining conditions. It might be desirable to also try solving these equations with a purely numerical solver. The following interpolant, one of a family of equivalent

interpolants, was calculated in this way, with free parameters identified during the solution process set to 0 at the end. It should be noted that an alternative procedure for setting free parameters would be to use them to minimize the interpolant error, yielded by the sixth degree term in z in the Taylor expansion of Equation 35.

$$\gamma_0(\theta)=\begin{bmatrix} 1+6.6257606574898842\theta-2.4954467276874458\theta^2-10.5065833728538 6\theta^3+10.444163321165534\theta^4-5.3085894936524999\theta^5 \\ 1.2604945758471145\theta \\ -2.4812693924523270\theta \\ -2.9781439048957180\theta+4.8980509131990130\theta^2 \\ -2.4268419359889530\theta-2.4026041855115560\theta^2+10.5065833728538 6\theta^3-10.444163321165534\theta^4+5.3085894936524999\theta^5 \end{bmatrix}^T$$

$$\beta_0(0)=\begin{bmatrix} -14.438897214071690\theta+7.1519733206074640\theta^2+21.872934599169840\theta^3-20.678083571272680\theta^4+9.2550967799316 50\theta^5 \\ -7.2412112973764320\theta+4.5177685093791980\theta^2+1.0323260599151780\theta^3+3.6654559741325980\theta^5 \\ 6.0067479405540020\theta-6.8171730606096 30\theta^4 \\ -1.6721559444299640\theta-3.5943820301311530\theta^3+9.8691828436750 40\theta^4-4.0617228503119060\theta^5 \\ 0.1622576215384850\theta-0.2672115582600823 0\theta^2+1.6272657376669 10\theta^3-3.1876761861315 80\theta^4+1.7204430393825870\theta^5 \end{bmatrix}^T$$

Although there seemed to be only 10 free parameters based on the number of equations to solve, some of the equations proved to be dependent as they became effectively 0 before they could be utilized in the solution process.

The error coefficient is a point of comparison among competing methods as the coefficient of the leading discretization error term, for these DIMSIMs always multiplying a term of the form $y^{(p+1)}(t_n)h_n^{p+1}$. We note, first of all, that the error coefficient of any DIMSIM is given by

$$v^T\varphi_p=v^T\left(\sum_{k=1}^{p+1}\frac{\alpha_{p+1-k}}{k!}-\frac{Bc^p}{p!}\right),$$

from Theorem 6. For this method we find that

$$\varphi_p=\begin{bmatrix} 0.0000551866764036239 59 \\ 0.0001353800900320961 82 \\ 0.000450115705599051 38 \\ 0.001413135519993483 04 \\ -0.000673428518845803 9 \end{bmatrix}$$

and so the error coefficient of the method becomes $\nu^T \varphi_P$ = 1.38889e-3. However, interestingly enough, for the first step the comparable number obtained using Theorem 8 is 5.518667640362375e-05.

Error estimation parameters were also calculated. The initial error is estimated using vectors β and γ where

$$\beta = \begin{bmatrix} 0.1812875545254936 \\ 0.3105254335256836 \\ -0.3258295105315199 \\ 0.1138267699195702 \\ -0.01051030018603054 \end{bmatrix}$$

and

$$\gamma = \begin{bmatrix} -0.1351324309632461 \\ 0 \\ 0 \\ 0 \\ 0.1351324309632461 \end{bmatrix}.$$

These are calculated using the conditions of Theorem 8, which yield 7 linear equations for the 10 parameters. These were solved using Mathematica, and γ2, γ3, and γ4 were free parameters that were arbitrarily set to 0 to produce a simple form minimizing the number of operations required per step.

Subsequent steps utilize different vectors β and γ where these are determined using Theorem 9. Upon examination, the first condition of Equation 57 reduces in the case to the second and so there are again 7 linear equations in 10 unknowns, solved using Mathematica. For this method β and γ were derived to be

$$\beta = k \begin{bmatrix} -126.7321008977760 \\ -9.065592286799085 \\ 9.939847085378934 \\ -1.234330257611486 \\ 0 \end{bmatrix}$$

and

$$\gamma = k\begin{bmatrix} 67.38322836913306 \\ 0 \\ -65.05966877019942 \\ 0 \\ -2.323559598933673 \end{bmatrix},$$

where

$$k = \frac{\delta^4}{1 + 5.737741328958135\delta + 7.613785314977576\delta^2 + 2.929172473396786\delta^3 + .05312848737725841\delta^4}.$$

Observe that k has four poles, at δ = -52.4401, δ = -1.44533, δ = -1, and δ= -0.248337, and these negative poles are not a problem since δ must be positive in any ODE solver.

However, it should be noted that there was a choice here of free parameters, which were all arbitrarily set to 0 at the end of the calculation. These were β_5, γ_2 and γ_4. Other choices of free parameters were tried first and actually yielded poles for small positive δ. In particular, a choice of β_5, γ_3 and γ_4 yielded a pole at δ = 0.93028; a choice of β_5, γ_2 and γ_3 yielded a pole at δ = 0.0752006; and a choice of γ_2, γ_3 and γ_4 yielded a pole at δ = 0.568585. Use of free parameters to enforce the condition $\beta^T e=0$ also resulted in a small positive pole and so this condition was not enforced. Testing showed that so long as the pole was avoided, similar results were obtained for all choices of free parameters. Clearly it is undesirable for a pole to appear in the error estimation expression in the neighborhood of likely choices for successive step size ratios, and it is evident that these poles are unfortunate artifacts that do not reflect the actual errors produced through step-size changes. This problem was earlier observed by Jackiewicz and Zennaro (Reference 28) and by Bellen, Jackiewicz and Zennaro (Reference 29) in the context of two-step Runge-Kutta methods. The success here in using the parameter freedom to move these poles to the negative real axis is an encouraging result that may have direct relevance to other ODE methods.

For our explicit fifth order solver, we need a total of 5 equations to obtain for our 5 unknowns the derivatives 2 through 6. We use an order 6 explicit Runge-Kutta solver to obtain an approximation y_1 to $y(t_0+h_0)$ accurate to $O(h_0^7)$. Then we have:

$$y_1 = y_0 + h_0 y_0' + \tfrac{h_0^2}{2} y_0'' + \tfrac{h_0^3}{6} y_0''' + \tfrac{h_0^4}{24} y_0^{(4)} + \tfrac{h_0^5}{120} y_0^{(5)} + \tfrac{h_0^6}{720} y_0^{(6)} + O\left(h_0^7\right)$$
$$y_1' = y_0' + h_0 y_0'' + \tfrac{h_0^2}{2} y_0''' + \tfrac{h_0^3}{6} y_0^{(4)} + \tfrac{h_0^4}{24} y_0^{(5)} + \tfrac{h_0^5}{120} y_0^{(6)} + O\left(h_0^6\right).$$

Here $y_1' = f(t_0 + h_0, y_1)$. Similarly we obtain two more equations by computing an approximation y_2 to $y(t_0+h_0/2)$ accurate to $O(h_0^7)$. This yields

$$y_2 = y_0 + \frac{h_0}{2}y_0' + \frac{h_0^2}{8}y_0'' + \frac{h_0^3}{48}y_0''' + \frac{h_0^4}{384}y_0^{(4)} + \frac{h_0^5}{3840}y_0^{(5)} + \frac{h_0^6}{46080}y_0^{(6)} + O(h_0^7)$$

$$y_2' = y_0' + \frac{h_0}{2}y_0'' + \frac{h_0^2}{8}y_0''' + \frac{h_0^3}{48}y_0^{(4)} + \frac{h_0^4}{384}y_0^{(5)} + \frac{h_0^5}{3840}y_0^{(6)} + O(h_0^6).$$

Note that the 6th derivative term must be included to obtain sufficient accuracy, even if an a priori error estimate is not sought. This means that one more equation must be used, and this may come from computing an approximation y_4 to $y(t_0+h_0/4)$ accurate to $O(h_0^7)$. This final equation is

$$y_4 = y_0 + \frac{h_0}{4}y_0' + \frac{h_0^2}{32}y_0'' + \frac{h_0^3}{384}y_0''' + \frac{h_0^4}{6144}y_0^{(4)} + \frac{h_0^5}{122880}y_0^{(5)} + \frac{h_0^6}{2949120}y_0^{(6)} + O(h_0^7).$$

Eliminating variables we solve these five simultaneous equations and obtain

$$y_0'' = \frac{2}{9h_0^2}\left(-567y_0 - 25y_1 - 432y_2 + 1024y_4 + h_0(-90y_0' + 3y_1' + 72y_2')\right) + O(h_0^5),$$

$$y_0''' = \frac{1}{h_0^3}\left(1836y_0 + 148y_1 + 2112y_2 - 4096y_4 + h_0(222y_0' - 18y_1' - 384y_2')\right) + O(h_0^4)$$

$$y_0^{(4)} = \frac{32}{3h_0^4}\left(-1323y_0 - 169y_1 - 1836y_2 + 3328y_4 + h_0(-144y_0' + 21y_1' + 378y_2')\right) + O(h_0^3)$$

$$y_0^{(5)} = \frac{160}{h_0^5}\left(378y_0 + 70y_1 + 576y_2 - 1024y_4 + h_0(39y_0' - 9y_1' - 132y_2')\right) + O(h_0^2)$$

$$y_0^{(6)} = \frac{1280}{h_0^6}\left(-90y_0 - 22y_1 - 144y_2 + 256y_4 + h_0(-9y_0' + 3y_1' + 36y_2')\right) + O(h_0).$$

The error in the first step is given by Theorem 8, with h_1 the first DIMSIM step size, to be

$$lte_1 = \left(\frac{1}{6!} - \sum_{j=1}^{5} B_{1j} \frac{c_j^5}{5!} \right) y^{(6)}(t_0) + O\left(h_1^7\right)$$

$$= (7.063894579663840e-2)\delta^6\left(-90y_0 - 22y_1 - 144y_2 + 256y_4 + h_0\left(-9y_0' + 3y_1' + 36y_2'\right)\right)$$

$$+ O\left(h_1^7\right)$$

If we set this error to half the tolerance τ and use a norm to include the possibility of systems of equations, we then may write a conservative but hopefully accurate choice for initial step size to be δh_0, where

$$\delta = \left(\frac{\tau}{2(7.063894579663840e-02)\left\| -90y_0 - 22y_1 - 144y_2 + 256y_4 + h_0\left(-9y_0' + 3y_1' + 36y_2'\right) \right\|} \right)^{\frac{1}{6}}$$

TESTING IMPLEMENTATION PARAMETERS FOR FIFTH ORDER

a. Variable Step Error Estimate

The error estimate was tested in the same way for fifth order as for second order. For the same time steps the following results were obtained in Figure 5.

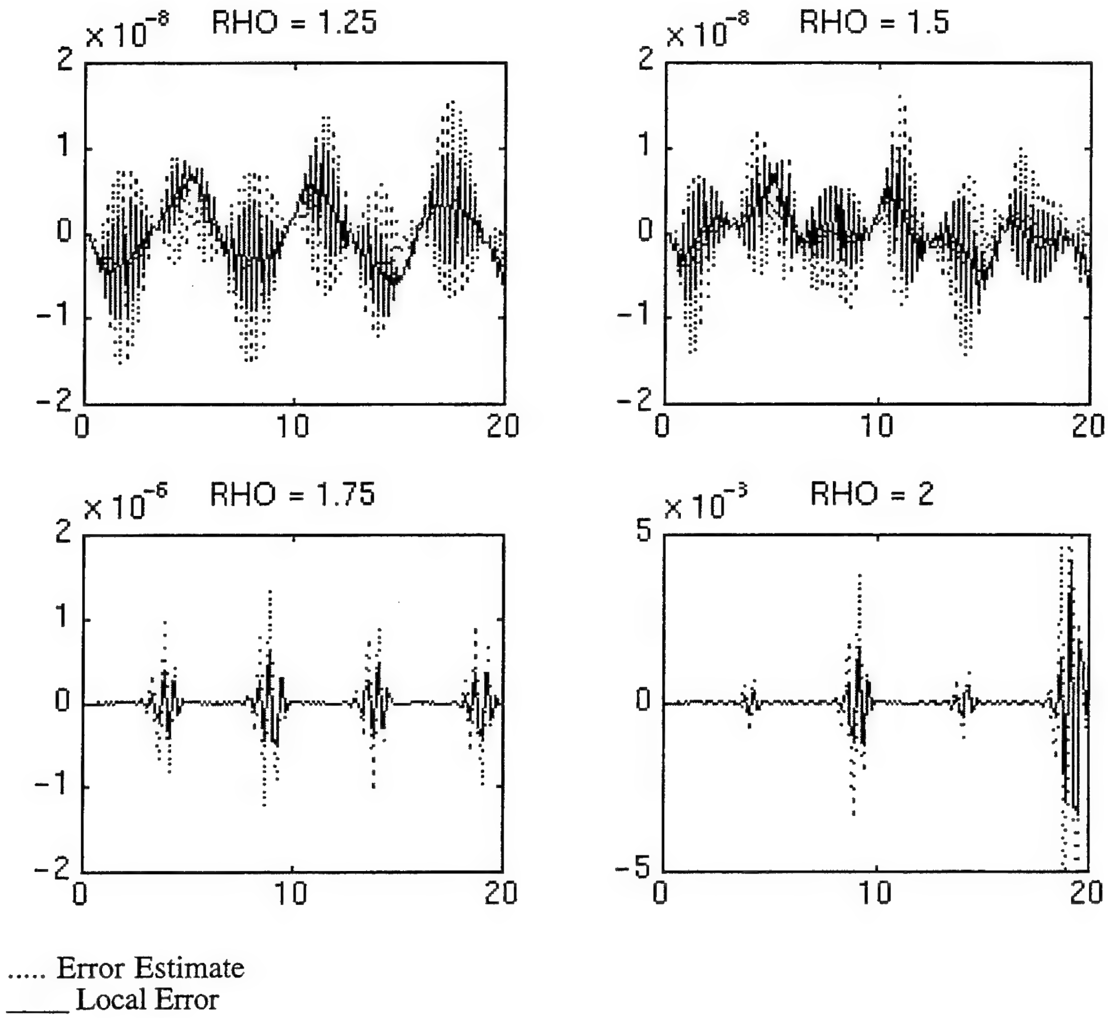


FIGURE 5. Order 5 Error Estimation Test.

The statistics obtained during the fifth order error stimation test are contained in Tables 9 and 10.

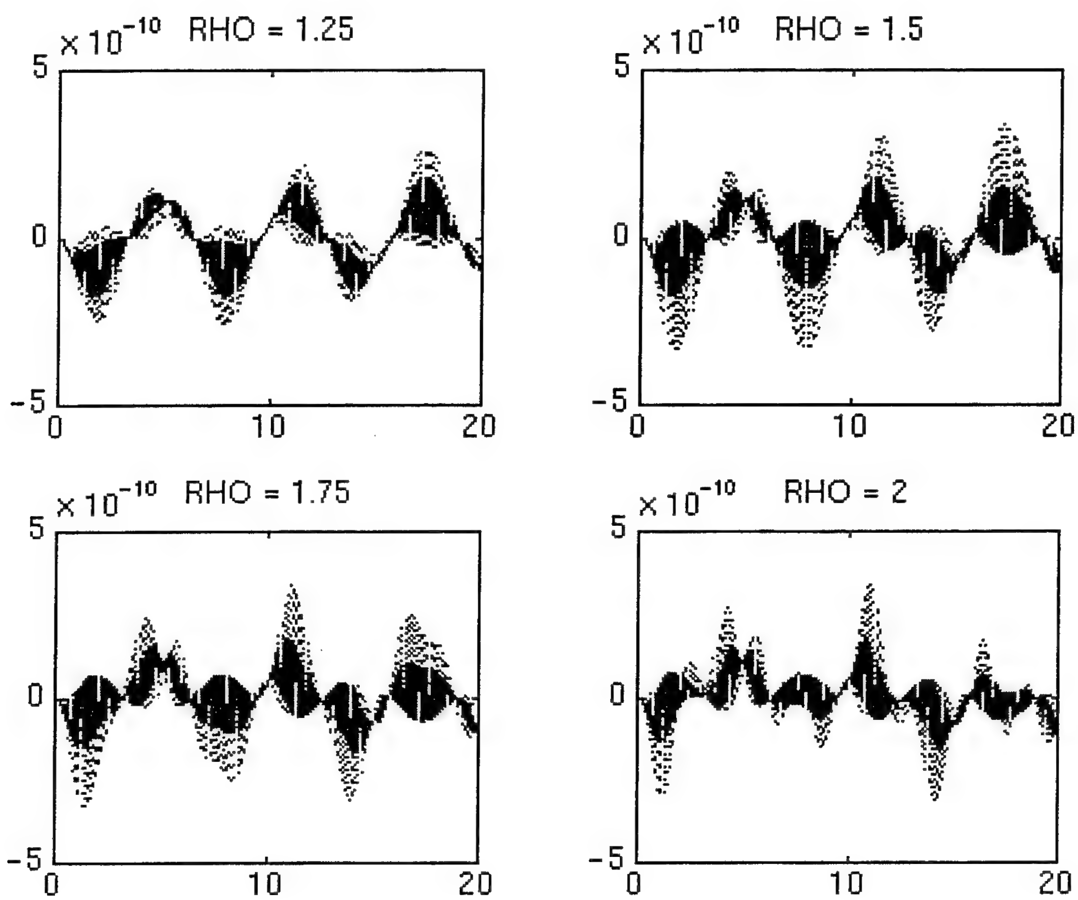
TABLE 9. Order 5 Error Estimation Test Part 1.

ρ	% r<.01	%r<.05	%r<.10	%r<.25	%r<.50	%r<1.0
1.25	0.66	3.31	6.62	17.88	52.32	88.08
1.50	1.32	8.55	15.13	29.61	54.61	94.08
1.75	0.66	3.95	8.55	16.45	38.16	90.79
2.00	0.00	0.65	5.88	11.11	25.49	77.77

TABLE 10. Order 5 Error Estimation Test Part 2.

ρ	rmin	rmax	t_f	Max err	Max err/Δt^5
1.25	2.74e-2	16.85	20.0402	1.54e-8	3.78e-4
1.50	2.99e-2	96.67	20.0963	1.59e-8	4.02e-4
1.75	1.76e-2	12.96	20.0335	1.50e-8	3.67e-4
2.00	5.84e-2	26.48	20.1128	3.21e-5	7.80e-1

The error here is significantly less than what appeared with order 2, but the problem for $\rho = 2$ with larger errors and change in solution order is troubling. A separate test was run with smaller step size, a difference of a factor of two. The results can be seen in Figure 6, and Tables 11 and 12.



..... Error Estimate
____ Local Error

FIGURE 6. Order 5 Error Estimation Test, Smaller Step Size.

TABLE 11. Order 5 Error Test, Smaller Step Size, Part 1.

ρ	% r<.01	%r<.05	%r<.10	%r<.25	%r<.50	%r<1.0
1.25	0.66	4.98	9.97	27.24	61.46	80.73
1.50	1.65	9.27	15.56	29.14	58.28	90.07
1.75	0.33	4.30	7.28	18.87	40.40	92.72
2.00	0.66	2.97	5.61	13.20	33.66	74.59

TABLE 12. Order 5 Error Test, Smaller Step Size, Part 2.

ρ	rmin	rmax	t_f	Max err	Max err/Δt^5
1.25	6.55e-4	490.6	20.0200	4.96e-10	3.83e-4
1.50	2.83e-2	73.26	20.0483	5.10e-10	3.99e-4
1.75	1.31e-2	41.03	20.0165	4.88e-10	3.76e-4
2.00	7.15e-3	35.51	20.0556	5.11e-10	3.92e-4

We can see that in this case the results are consistent for all ρ.

b. Interpolants

Interpolation testing was similar to the procedures used for the second order method, but with more points and over a shorter interval. The results for the Butcher-Jackiewicz interpolant are seen in Figure 7.

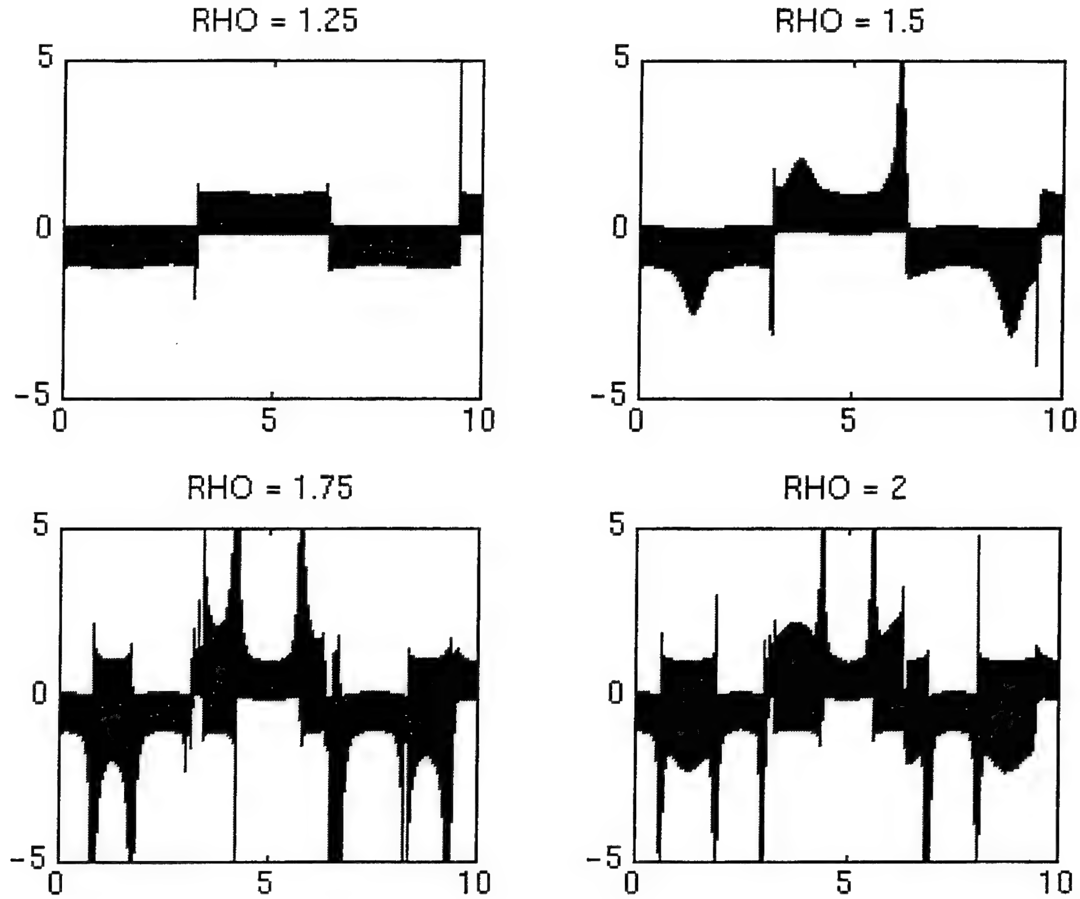


FIGURE 7. Test Results for Order 5 Butcher-Jackiewicz-Type Interpolant.

The statistics obtained during the fifth order interpolant test are contained in Table 13.

TABLE 13. Order 5 Butcher-Jackiewicz Interpolant Test.

ρ	$k > 10$	$10 > k > 5$	$5 > k > 2$	$2 > k > 1$	$k <= 1$
1.25	0.0%	0.0%	0.4%	8.8%	90.8%
1.50	0.2%	0.2%	3.5%	20.9%	75.1%
1.75	2.4%	2.0%	7.5%	28.1%	59.9%
2.00	1.2%	0.8%	6.5%	32.6%	58.8%

Evidently some significant degradation occurs at this higher order with increasingly rapid step changing. A similar phenomenon was also observed in a study of related general linear methods by Enenkel and Jackson (Reference 30) and Enenkel (Reference 21). It should be again noted that no error minimization was carried out using the free parameters.

The simple fifth order Nordsieck interpolant yielded the following in Figure 8.

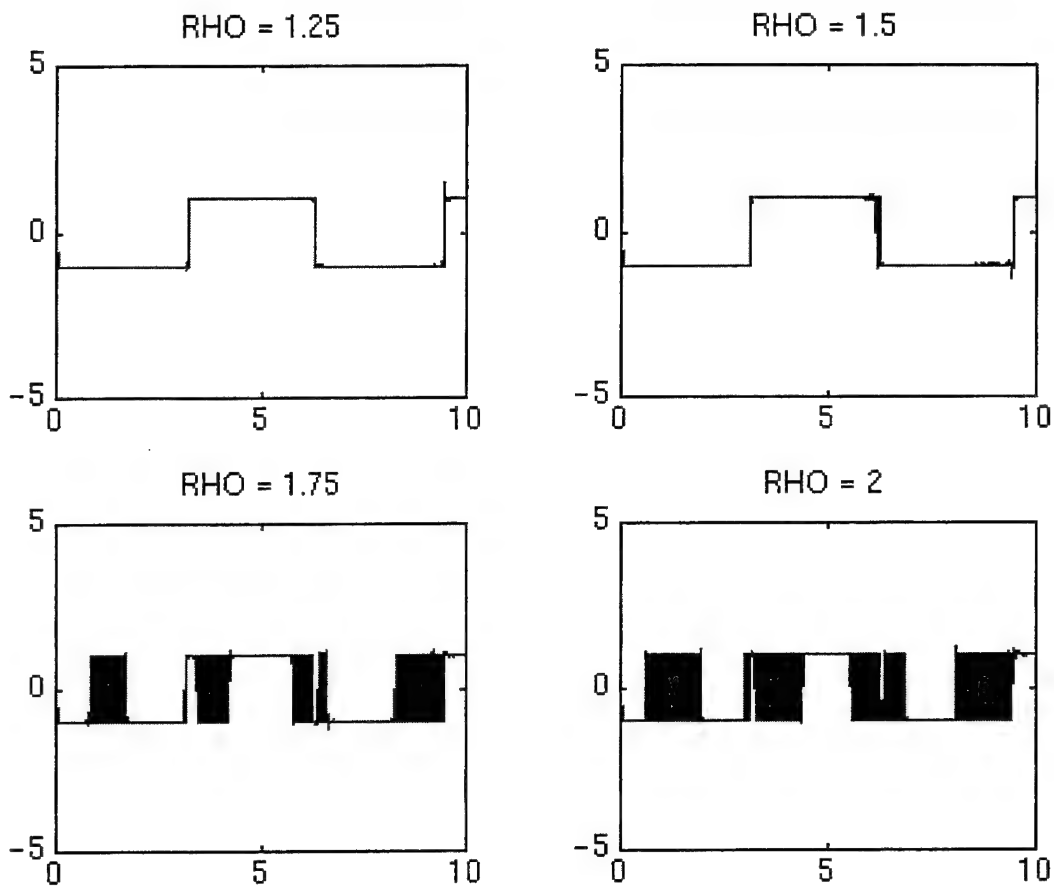


FIGURE 8. Order 5 Nordsieck Interpolant.

The statistics obtained during the fifth order Nordsieck Interpolant test are contained in Table 14.

TABLE 14. Order 5 Nordsieck Interpolant.

ρ	k > 10	10 > k > 5	5 > k > 2	2 > k > 1	k <= 1
1.25	0.0%	0.0%	0.0%	88.5%	11.5%
1.50	0.0%	0.0%	0.0%	86.5%	13.5%
1.75	0.0%	0.0%	0.1%	84.5%	15.4%
2.00	0.0%	0.0%	0.0%	85.5%	14.5%

This is clearly highly accurate for this range of step sizes.

The continuous Nordsieck interpolant displayed even greater accuracy in Figure 9.

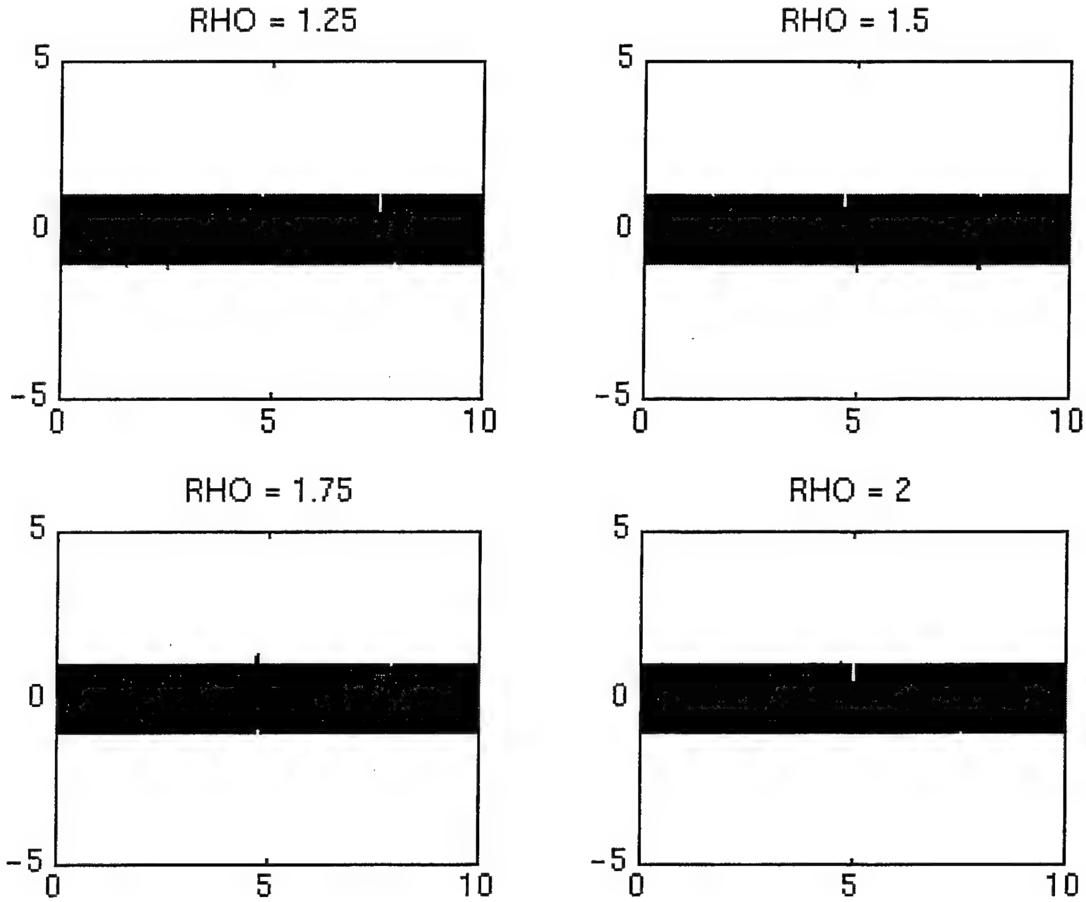


FIGURE 9. Order 5 Continuous Nordsieck Interpolant.

The statistics obtained during the fifth order continuous Nordsieck interpolant test are included in Table 15.

TABLE 15. Order 5 Continuous Nordsieck Interpolant.

ρ	k > 10	10 > k > 5	5 > k > 2	2 > k > 1	k <= 1
1.25	0.0%	0.0%	0.0%	1.1%	98.9%
1.50	0.0%	0.0%	0.0%	1.3%	98.7%
1.75	0.0%	0.0%	0.0%	1.4%	98.6%
2.00	0.0%	0.0%	0.0%	1.3%	98.7%

This must be considered highly desirable, perhaps with more accuracy than is typically needed.

It then became of interest to test the results for the larger step sizes used for 2nd order. The results for the Butcher-Jackiewicz interpolant were as follows in Figure 10.

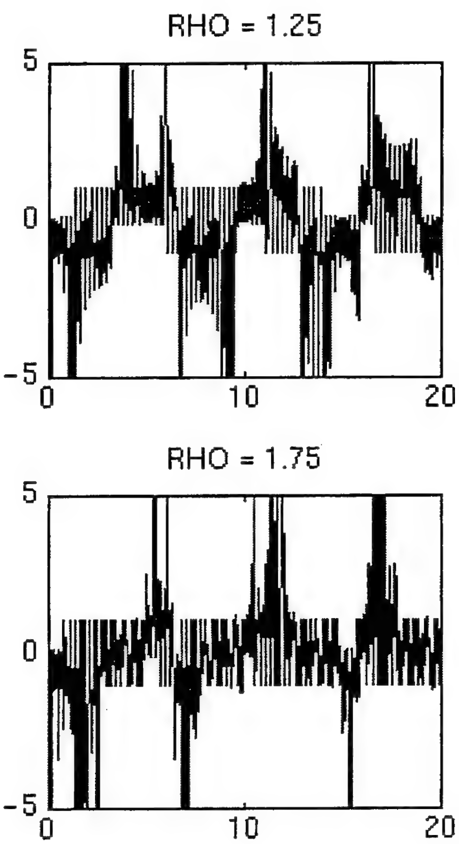


FIGURE 10. Order 5 Butcher-Jackiewicz Interpolant Test, Longer Step Size.

The statistics obtained during the fifth order Butcher-Jackiewicz interpolant test are contained in Table 16.

TABLE 16. Order 5 Butcher-Jackiewicz Interpolant Test, Longer Step Size.

ρ	k > 10	10 > k > 5	5 > k > 2	2 > k > 1	k <= 1
1.25	3.9%	3.4%	13.4%	26.4%	52.9%
1.50	3.4%	3.8%	13.9%	33.9%	45.1%
1.75	3.6%	3.4%	6.8%	24.7%	61.5%
2.00	2.5%	2.5%	4.5%	21.9%	68.6%

For the fifth order Nordsieck interpolant the results are shown in Figure 11.

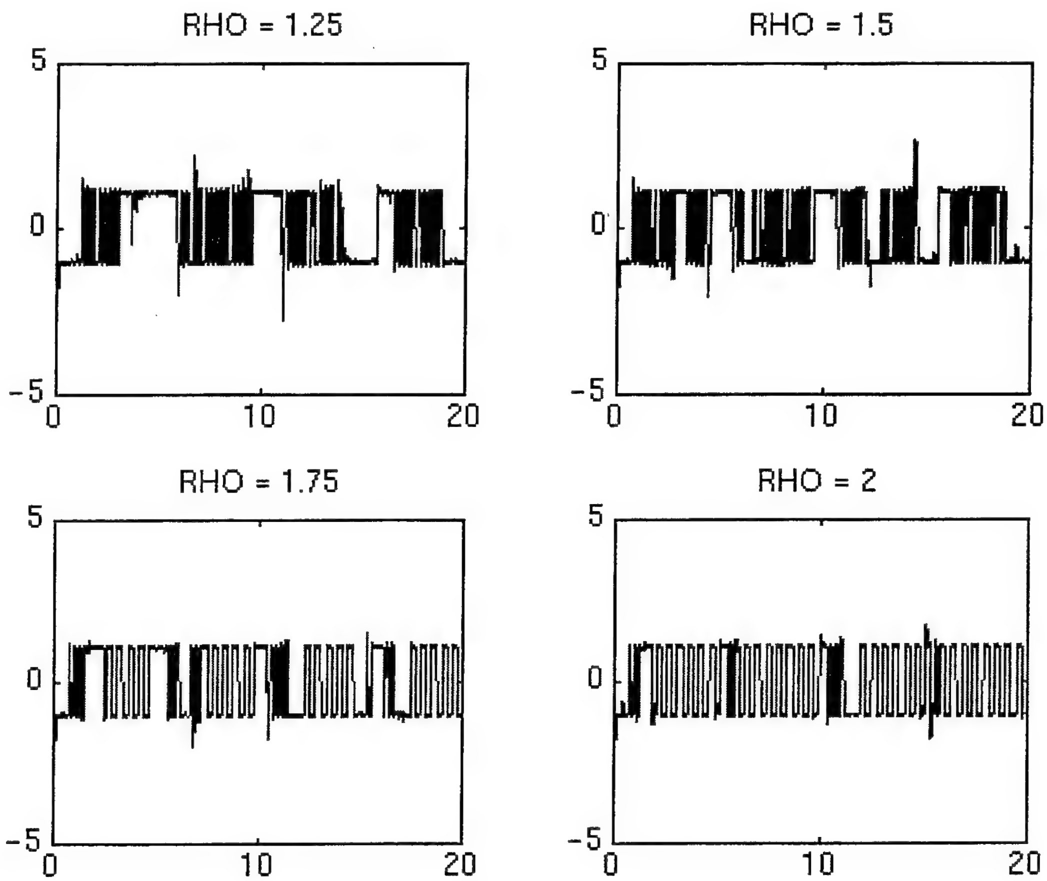


FIGURE 11. Nordsieck Interpolant Test, Longer Step Size.

The statistics obtained during the Nordsieck interpolant test are contained in Table 17.

TABLE 17. Nordsieck Interpolant Test, Longer Step Size...

ρ	k > 10	10 > k > 5	5 > k > 2	2 > k > 1	k <= 1
1.25	0.0%	0.0%	0.5%	80.7%	18.8%
1.50	0.0%	0.0%	0.3%	83.0%	16.7%
1.75	0.0%	0.0%	0.3%	86.8%	12.9%
2.00	0.0%	0.0%	0.0%	88.8%	11.2%

For the continuous Nordsieck interpolant we have Figure 12 and Table 18.

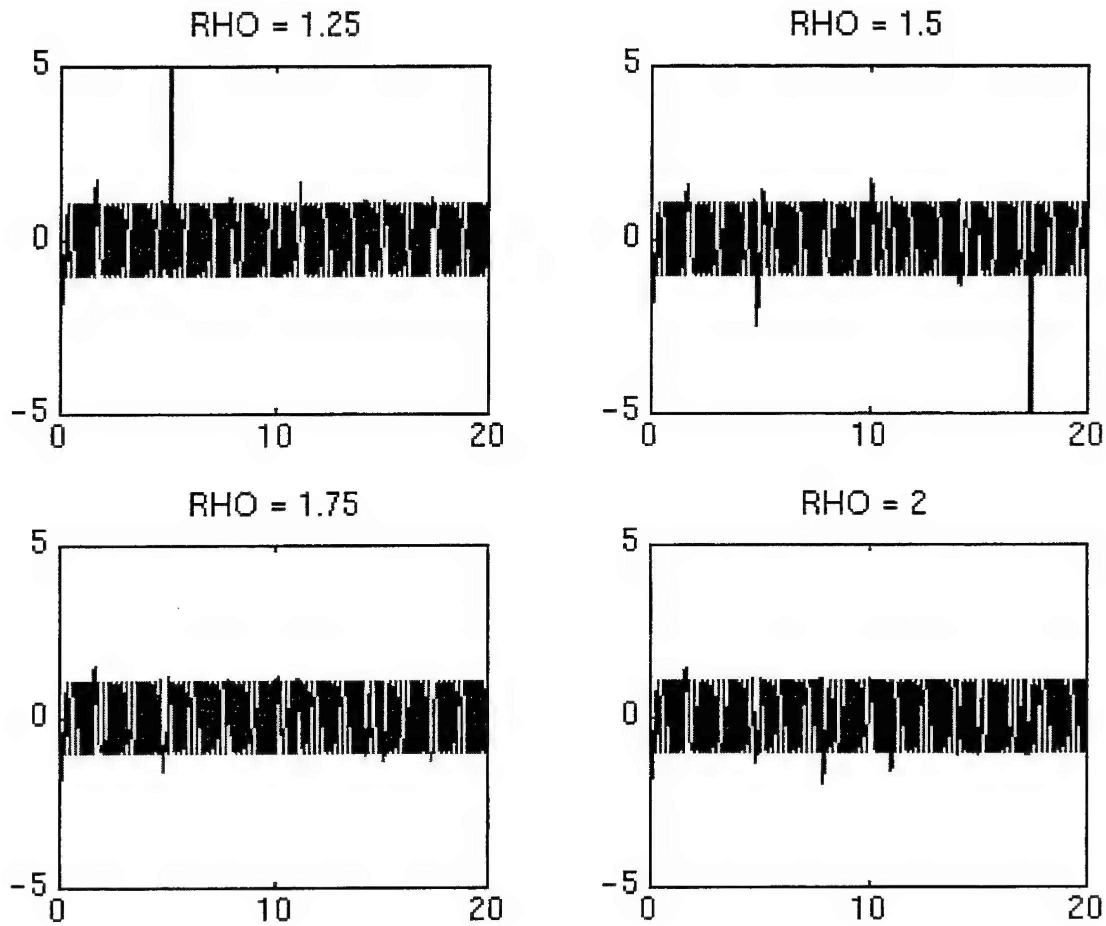


FIGURE 12. Order 5 Continuous Nordsieck Interpolant Test, Longer Step Size.

TABLE 18. Order 5 Continuous Nordsieck Interpolant Test, Longer Step Size.

ρ	f > 10	10 > f > 5	5 > f > 2	2 > f > 1	f <= 1
1.25	0.5%	0.1%	0.0%	25.8%	73.6%
1.50	0.5%	0.1%	0.3%	26.7%	72.4%
1.75	0.0%	0.0%	0.0%	26.5%	73.5%
2.00	0.0%	0.0%	0.0%	26.3%	73.7%

Both Nordsieck interpolants seem to be highly acceptable, with the simple Nordsieck interpolant producing fewer large errors with $k > 5$, while the continuous Nordsieck interpolant otherwise generating higher accuracy. The continuous Nordsieck interpolant seems to consistently produce $k < 1$, while the simple Nordsieck interpolant produces many values of k slightly greater than 1.

The starting procedure testing was similar to that used for the second order method. The selection of h_0 proved to be a bit delicate and, as alluded to above, the formula $gh_0^6 = 10^5 eps$ proved to be necessary to provide a suitable value. If the eps factor was reduced by a factor of 10 the step was too small to prevent serious round-off errors from occuring, while if the factor was increased by 10 there was a loss of local truncation accuracy. This is not because of the breakdown in the assumption that the derivatives are of the same size as the initial value and the corresponding derivative. In fact in this test 6th derivatives grew to be as large as 64 while first derivatives and functional values were around 0.3 to 1, but this would tend to reduce rather than increase the size of h_0. A better answer is that the amount of cancellation in the calculation of the Nordsieck vector components requires that more digits exist in the answer.

The testing of the starting method concerns the accuracy of the starting vector, the accuracy of the first step error estimate, and the appropriateness of the choice for initial step size. The Prothero-Robinson test problem is again used here with $\lambda = -2$, with a starting point at t_0 on the exact solution trajectory through (0,1).

The following test (Table 19) reveals the accuracy of the error estimate and of the initial step size selection algorithm. The starting point at $t_0 = 1$ is used here throughout. It is evident that accuracy improves until round-off error becomes significant.

TABLE 19. Order 5 Starting Procedure Test 1.

t_0	h (Nord)	Loc err	Err est	h (H-N-W)	Loc err	Err est
10^{-3}	1.03	-5.22×10^{-5}	6.81×10^{-4}	1.76×10^{-1}	8.05×10^{-9}	1.34×10^{-8}
10^{-6}	3.26×10^{-1}	2.12×10^{-7}	5.77×10^{-7}	5.55×10^{-2}	1.09×10^{-11}	1.26×10^{-11}
10^{-9}	1.03×10^{-1}	3.96×10^{-10}	5.32×10^{-10}	1.76×10^{-2}	1.90×10^{-14}	1.12×10^{-14}
10^{-12}	3.26×10^{-2}	5.09×10^{-13}	5.00×10^{-13}	5.55×10^{-3}	5.55×10^{-16}	-2.39×10^{-17}
10^{-15}	1.03×10^{-2}	1.89×10^{-15}	3.26×10^{-16}	1.76×10^{-3}	5.55×10^{-16}	-3.33×10^{-18}

Another revealing test uses starting points at 0 through 1, evenly spaced, and with a fixed tolerance of 10^{-9}. At this tolerance and at this order the error estimate is not quite as accurate. We note again that the use of the Nordsieck-vector-based error estimate for the first step provides a much more efficient start than the method outlined in Hairer, Norsett and Wanner (Table 20).

TABLE 20. Order 5 Starting Procedure Test 2.

t_0	h (Nord)	Loc err	Err est	h (H-N-W)	Loc err	Err est
0	7.22×10^{-2}	4.23×10^{-10}	5.16×10^{-10}	1.17×10^{-2}	8.77×10^{-15}	8.71×10^{-15}
0.2	7.75×10^{-2}	4.29×10^{-10}	5.26×10^{-10}	1.26×10^{-2}	1.39×10^{-14}	9.59×10^{-15}
0.4	8.32×10^{-2}	4.16×10^{-10}	5.28×10^{-10}	1.45×10^{-2}	1.20×10^{-14}	1.48×10^{-14}
0.6	8.91×10^{-2}	4.12×10^{-10}	5.31×10^{-10}	1.60×10^{-2}	1.65×10^{-14}	1.71×10^{-14}
0.8	9.54×10^{-2}	4.14×10^{-10}	5.28×10^{-10}	1.80×10^{-2}	2.93×10^{-14}	2.36×10^{-14}
1	1.03×10^{-1}	3.97×10^{-10}	5.32×10^{-10}	1.76×10^{-2}	1.90×10^{-14}	1.12×10^{-12}

The final test concerns the calculation of the starting vector. Here we are concerned that the smallest possible safe value for h_0 be utilized to produce the maximum accuracy since no DIMSIM integration step will actually be taken until this is rescaled. As indicated above, we use a choice of $h_0 = 10^{\frac{5}{p+1}}\left(\frac{eps}{g(t_0, y_0)}\right)^{\frac{1}{p+1}}$, with p = 5. We test this by calculating the relative errors in the fifth and sixth derivatives (noting that higher derivatives are computed less accurately) at 5 uniformly spaced starting points between 0 and 1. The error in the starting vector is also calculated (Table 21).

TABLE 21. Order 5 Starting Procedure Test 3.

t_0	h_0	$y^{(5)}(t_0)$	Rel err	$y^{(6)}(t_0)$	Rel err	$\left\|\Delta y^{[0]}\right\|_\infty$
0	$1.68\text{x}10^{-2}$	$-3.10\text{x}10^{1}$	$1.13\text{x}10^{-5}$	$6.40\text{x}10^{1}$	$2.75\text{x}10^{-3}$	$4.51\text{x}10^{-15}$
0.2	$1.99\text{x}10^{-2}$	$2.05\ \text{x}10^{1}$	$1.47\text{x}10^{-4}$	$4.27\ \text{x}10^{1}$	$2.29\text{x}10^{-2}$	$1.72\ \text{x}10^{-14}$
0.4	$3.16\text{x}10^{-2}$	$-1.35\ \text{x}10^{1}$	$6.31\text{x}10^{-4}$	$2.84\ \text{x}10^{1}$	$3.81\text{x}10^{-2}$	$4.48\ \text{x}10^{-13}$
0.6	$2.15\text{x}10^{-2}$	-8.81	$3.74\text{x}10^{-4}$	$1.87\ \text{x}10^{1}$	$3.44\text{x}10^{-2}$	$1.76\ \text{x}10^{-14}$
0.8	$2.06\text{x}10^{-2}$	-5.76	$2.59\text{x}10^{-5}$	$1.22\ \text{x}10^{1}$	$1.61\text{x}10^{-2}$	$1.48\ \text{x}10^{-14}$
1	$2.09\text{x}10^{-2}$	-3.79	6.64x10-^{4}	7.82	$3.86\text{x}10^{-2}$	$2.48\ \text{x}10^{-14}$

THE DIMEX FAMILY OF EXPLICIT DIMSIM ODE SOLVERS

DESCRIPTION AND USE OF DIMEX FAMILY OF SOLVERS

The second order type 1 DIMSIM explicit variable step-size ordinary differential equation-initial value problem solvers DIMEXx are written as a collection of double precision FORTRAN 77 subroutines called by a driver dimx1 that provides the interface to the user's calling program. Here x denotes the method order. These are research codes, designed primarily for use in waveform relaxation in a form that should make them suitable for extension to higher orders, as well as for use in solving standard ODE initial value problems. A number of elements remain to be determined in developing mature production codes, and optimization for speed and memory usage have not been performed. The codes use the Type 1 schemes along with the coefficients described in the previous chapter. Advantage is taken of the FASAL property referred to in the Implementing DIMSIMs section whenever use of the same step size is continued for another step for both order 2 and order 5, and for all steps after the first for second order. In this section we discuss other significant elements of the software design and use. The complete code for DIMEX5 appears in the appendix.

a. Calling the Solvers

The user must issue the command (x is the order number, currently 2 or 5)

```
call dimx1(X0,Y0,X,Y,F,NEqn,XF,H,ATol,RTol,Starting,XOut,
    1      NCalls,NMisses,Diag,ErrEst,Work)
```

Parameter definitions:

X0: (real*8) Initial value of independent variable at start of integration, input

Y0: (real*8 array) Initial value of solution at start of integration, dimensioned NEqn, input

X: (real*8) Desired output point if Diag is .false., otherwise set to same as XF, input

Y: (real*8) Computed solution at X if Diag is .false., otherwise at XOut, output only

F: (subroutine) Provides derivative function f for ODE y'=f(x,y). Subroutine must be defined with parameters F(X,Y,YP,NEqn), where YP is the derivative at (X,Y) and NEqn is as defined below. F must be declared external in the calling program.

NEqn: (integer) Number of equations in ODE system, input

XF: (real*8) Intended termination point for integration process, input

H: (real*8) Step size used. Output only, not to be defined or changed by the user

ATol: (real*8) Absolute tolerance for local truncation error, used in controlling integration according to formula described below, input

RTol: (real*8) Relative tolerance for local truncation error, used in controlling integration according to formula ErrEst<ATol+RTol* $\|Y\|_\infty$, input

Starting: (logical) Initially set to .true., thereafter not to be changed by the user

XOut:: (real*8) End point of last integration step performed, output only, not to be changed by the user

NCalls: (integer) Number of function calls since beginning of integration. Output only, not to be changed by the user.

NMisses: (integer) Number of tolerance misses since beginning of integration. Output only, not to be changed by the user.

Diag: (logical) Set to true if only one integration step at a time is desired, which is the preferred mode for study of integrator behavior, and false if it is desired to integrate until interpolated output at X may be obtained

ErrEst: (real*8) Maximum norm of estimated error in last integration step taken, output only

Work: (real*8) Work array which must be dimensioned at least 2+24*NEqn for 2nd order and 2+53*NEqn for 5th order.

b. Calling tree for DIMEX5

```
                               dim51
                                 |
                               driver
                                 |
 ____________________________________________________________________
 |                       |              |             |            |
start                  solver        NewStep     NordInterp     rescale
 |     \                 |
 F     RK6               F
        |
        F
```

-A block data subprogram called matrices is also used.

FIGURE 13. Calling Tree for DIMEX5.

c. Description and parameters for subprograms of DIMEX5

subroutine dim51: See Section a above for discussion of parameters. This subroutine simply serves the purpose of hiding from the user the details of the numerous arrays used by DIMEX5. A single large work array is split appropriately as required by the driver.

```
subroutine driver(X0,Y0,X,Y,F,NEqn,XF,H,ATol,RTol,Starting,XOut,
    1      NCalls,NMisses,Diag,ErrEst,HOld,YIter,FStage,YIterP,
    2      XOld,YP,Y1,Y2,Y4,Y1P,Y2P,YStage,
    3      YIterN,YIterS,FStageN,Y6Der,FStageP,RKWork,NordVec)
```

driver provides the essential logic to control the integration process and calls the subprograms needed to carry out the necessary calculations. When Starting is set to .true., start is called to obtain an initial Nordsieck vector, step size, and external stage vector. Solver then is called to carry out the first integration step and the error estimate is checked to ensure the result is within tolerance. If not, the step size is halved and Nordsieck vector is rescaled to produce a new starting external stage vector. This process is repeated until successful. Starting is then set to .false. and the integration process proceeds as NewStep calculates a new step size to use. If Diag is .true. driver returns dim51, which returns to be called again by the main program, otherwise at each iteration either XOut is at least X, in which case NordInterp produces an interpolated solution value at X and a return is made to the main program for another call, or an integration step is carried out. An integration step begins by saving copies of the current and previous external stage vectors. Then rescale is called to rescale the current external stage vector, a DIMSIM step is performed using a call to solver, and ErrEst is checked to be sure the error is within tolerance. If not, a tolerance miss is counted and the step size is halved. Rescaling again takes place using the saved values, and this process continues until the step is within tolerance or MaxTries is exceeded, generating an error message and STOP. For normal execution final XOut is XF.

X0,Y0,X,Y,F,NEqn,XF,H,ATol,RTol,Starting,XOut,NCalls,NMisses,Diag,ErrEst: As in dim51, see Section a above.
HOld: (real*8) Step size used for previous successful integration step
YIter: (real*8 array) External stage vector for current step, dimensioned (NEqn,5)
FStage: (real*8 array) Vector of derivative values computed from internal stage values for most recent successful step, dimensioned (NEqn,5).
YIterP: (real*8 array) External stage vector for previous successful step, dimensioned (NEqn,5)
XOld: (real*8) Previous successful step point
YP: (real*8 array) Derivative at X0, =f(X0,Y0). Dimensioned NEqn.
Y1,Y2,Y4,Y1P,Y2P, Y6Der: (real*8 arrays) Storage needed for subroutine start. Each dimensioned NEqn.
YIterN,YIterS, FStageN: (real*8 arrays) Storage needed for temporary storage of current and previous external stage vectors and current vector of derivatives at internal stage points until it is verified that step was successful, dimensioned (NEqn,5).
YStage: (real*8 array) Storage needed for subroutine solver, dimensioned NEqn.
FStageP: (real*8 array) Previous vector of derivative values for last internal stage, used again for first internal stage of new step if step size does not change. Dimensioned NEqn)
RKWork: (real*8 array) Storage needed for subroutine start, dimensioned 8*NEqn.
NordVec: (real*8 array) At first step, Nordsieck vector at X0. Subsequently, Nordsieck vector at XOut. Dimensioned (NEqn,6)

```
subroutine solver(X0,NEqn,YIter,H,HOld,F,ErrEst,FStage,YIterP,Starting,YStage,
1                 FStageP,KCalls)
```

solver applies the basic DIMSIM algorithm to compute a new external stage vector and an error estimate. Different coefficients are used in computing the error estimate if Starting is .true.

X0: (real*8) Starting point for this integration step
NEqn,H,HOld,F,ErrEst,Starting: See description of driver, above.
FStage: (real*8 array) Upon output, set to vector of derivatives at internal stage points for current step, dimensioned (NEqn,5)
YIter: (real*8) Upon input, set to rescaled previous external stage vector. Upon output, set to current external stage vector. Dimensioned (NEqn,5)
YIterP: (real*8 array) Output only, set to input value of YIter. Dimensioned (NEqn,5)
YStage: (real*8 array) Temporary storage needed for each internal stage vector in turn during calculation. Dimensioned NEqn.
FStageP: (real*8 array) Input, see driver description above. Dimensioned NEqn.
KCalls: (integer) Number of calls to derivative function subroutine for this solver step, output

```
subroutine start(YIterP,F,X0,Y0,NEqn,H,YP,Tol,Y1,Y1P,Y6Der,Y2,Y2P,Y4,
1      YNord,RKWork)
```

Start first calculates the derivative at the initial point X0. A scaling factor g to estimate the size of the initial Nordsieck vector is computed and this is used to determine h_0. In most cases g will be the smaller of the norm of the function and the first derivative, but in case one is smaller than 1000*eps (machine epsilon) the larger is taken, and in no case is g taken as less than 10^{-6}. A 6th order Runge-Kutta method is used by calling RK6 to compute the solution Y4 at X0+h0/4, Y2 at X0+h0/2 (from Y4), and Y1 at X0+h0 (from Y2). These are then used to calculate a Nordsieck vector for computation of the initial external stage vector, plus an extra

derivative for computation of the appropriate size for the first DIMSIM step. Both the Nordsieck vector and the intial external stage vector are returned in case tolerance is not met.

YIterP: (real*8 array) Initial external stage vector, output dimensioned (NEqn,5)

F,X0,Y0,NEqn: See Section a above.

H: (real *8) Output value for calculated initial DIMSIM step size

YP: (real*8 array) Derivative of the solution at X0, calculated using y'=F(X0,Y0), output dimensioned NEqn

Tol: (real*8) Initial tolerance computed as described in section a from RTol, Y0 and ATol, input

Y1,Y1P,Y2,Y2P,Y4: (real*8 arrays) Values computed using 6th order Runge-Kutta method for solution at h_0, $h_0/2$, and $h_0/4$, and associated derivatives computed using F, workspace each dimensioned NEqn.

Y6Der: (real*8 array) 6th derivative of solution vector at X0, multiplied times h_0^6, workspace dimensioned NEqn.

YNord: (real*8 array) Nordsieck vector at X0 for step size H, output dimensioned (NEqn,6)

RKWork: See driver description above.

subroutine rescale(YIter,H,HOld,FStage,YIterP,NEqn)

Rescale takes a previous external stage vector computed using a step size HOld and uses the DIMSIM rescaling algorithm to rescale it to become the appropriate previous external stage vector for continuing the computation with step size H.

YIter: (real*8 array) Output rescaled external stage vector, dimensioned (NEqn,5)

H: (real*8) Input new step length

HOld: (real*8) Input step length for previous successful step.

FStage: (real*8 array) Input vector of derivatives at stage points for last step completed

YIterP: (real*8 array) Input previous external stage vector, calculated using step length HOld, dimensioned (NEqn,5)

NEqn: See Section a above

real*8 function NewStep(H,Tol,ErrEst)

The error estimate is compared with the tolerance to produce a step change factor modifying the step size in preparation for the next step. The local truncation error is assumed to be proportional to the 6th power of the step size. The largest possible step change is a factor of 2. A step change safety factor equal to .75 was used for DIMEX5 so that the step size used was .75 of optimal. A larger factor of .9 was used successfully for DIMEX2. Step size is not changed if change would be less than 10% for DIMEX5 and 5% for DIMEX2.

H: (real*8) At input contains old step size. At output contains new step size

Tol: (real*8) Input tolerance derived from RTol, Y and ATol as described in section a above

ErrEst: (real*8) Input current local error estimate

subroutine NordInterp(Y,H,Theta,FStage,YIter,NordVec,NEqn)

The Nordsieck vector NordVec is calculated at the current value of XOut, using the previously calculated external stage vector YIter, the current step size H, and the current vector of derivatives at stage points, FStage. Theta then is used to rescale the Nordsieck vector to correspond to the desired output point and a fifthth order Taylor series approximation Y is formed.

Y: (real*8 array) Output approximation to the solution at the desired output point, length NEqn.

H: (real*8) Input step size for last successful integration

Theta: (real*8) Input between 0 and 1, ratio (X-XOld)/H, where X is the desired output point

FStage: (real*8 array) Input vector of derivatives at stage points for last successful integration, dimensioned (NEqn,5)

YIter: (real*8 array) Input previous external stage vector, dimensioned (NEqn,5)

NordVec: (real*8 array) Workspace (output for waveform relaxation) for Nordsieck vector at current XOut and last step size used.

NEqn; (integer) See above, Section a.

subroutine RK6(Y,X0,Y0,H,F,NEqn,YP,YP2,YP3,YP4,YP5,YP6,YP7,YTemp)

A simple 6th order Runge-Kutta solver, used to provide solution values needed for starting method. Solution Y for initial values (X0,Y0) is found at X0+H for problem Y'=F(X,Y). The method is described in Butcher (Reference 11), pp. 203-205.

Y: (real*8 array) Solution at X0+H, dimensioned NEqn.

X0: (real*8) Initial value of independent variable.

Y0: (real*8 array) Initial value of dependent variable, dimensioned NEqn

H: (real*8) step size used

F: See above, Section a

NEqn: See above, Section a

YP,YP2,YP3,YP4,YP5,YP6,YP7: (real*8 arrays) Workspace used for storing derivatives of stages, each dimensioned NEqn.

YTemp: (real*8 array) Workspace used for storing stage vectors successfully, dimensioned NEqn

block data matrices

The matrices and vectors defining the paricular method are set to appropriate values using data statements and stored in the common region /Method/.

TESTING DIMEX

The ultimate test of an ODE solver is the accurate and efficient solution of a wide variety of problems. The relatively easy, nonstiff problems utilized in the well-known suite DETEST have provided a standard test suite for ODE solvers since they were first introduced in 1972 by Hull, et. al, (Reference 31). For second order testing these were run using both DIMEX2 and the well established linear multistep solver LSODE developed at Livermore by Hindmarsh and his coworkers (Reference 32). LSODE was used in the Adams-Moulton solver mode with functional iteration. This mode provides predictor-corrector solution of the form $P(EC)^M$ where M is no greater than 3 and is usually 1. This provided the most efficient solution for competitive purposes on this test suite. For fifth order testing these were run using both DIMEX5 and the highly regarded Runge-Kutta code DOPRI5 (Reference 18). For the easy problems of DETEST, LSODE is very efficient as measured by steps taken and function evaluations, but the sphere of applicability for DIMEX5 would be similar to that of DOPRI5 and it is hoped that a comparison on simpler problems would provide some indication of relative performance on the harder problems that do not find their way into standard benchmark suites. The problems are divided into classes A-E, with Class A consisting of single equations, Class

B of small systems of 2 or 3 equations, Class C of moderate systems of 10 to 51 equations, Class D of orbit equations, and Class E of higher order (second order) equations. Application areas represented include the simple negative exponential, a simple Riccati equation, an oscillatory problem, logistics and spiral curves, conflicting population growths, chemical reactions, rigid body motion, radioactive decay chains, heat equation, 5 body problem, orbital motion, Bessel's equation, Van der Pol's equation, Duffing's equation, falling body and linear pursuit. All are integrated over an interval from 0 to 20. Only a few are provided with exact solutions, and accuracy testing was conducted primarily using significantly tighter tolerances with the same solver. Rather than adapting DIMEX, DOPRI5 and LSODE to fit with the DETEST software, the functions were simply coded and used with a specially designed calling routine that provided the desired output information. Although number of steps is important, even more important is the number of function evaluations, which is a good measure of the amount of work required. A timing of the solution process would also give a measure of the overhead, which has been noted to be significant for LSODE (Reference 18). But the DIMEX codes have not been carefully optimized yet and so this comparison was not carried out.

Some aspects of the comparison process utilized should be noted. First, LSODE uses an Adams-Moulton solver and can only be set to restrict to a specified order up to 12 that will not be exceeded. Thus for second order testing it is possible that some steps were first order. The order is chosen to minimize error and hence maximizes step length. Furthermore, in conducting these tests it was observed that LSODE applies relative tolerance to each separate solution component, while DIMEX applies relative tolerance to the norm of the solution. Thus the only relative tolerance comparisons are for tests A1-A5, since only a single equation is used. Otherwise, absolute tolerance was used for both. Finally, exact solutions were used for comparison purposes only for A1-4, while high tolerances of 10^{-13} to 10^{-15} were used to produce a "true" value of the local solution and the end point solution.

For fifth order testing was conducted only for the more appropriate higher tolerances 10^{-6}, 10^{-9}, and 10^{-12}. On the other hand, second order methods often take a very long time to integrate at tight tolerances and so 10^{-12} was not used for second order, while the loose tolerance of 10^{-3} was an appropriate condition for testing a second order solver.

A summary of results is shown in the Tables 22 through 71. Note that an entry of "**" indicates that a measurement was not made. It was not convenient to obtain data from LSODE on tolerance misses and times deceived, and it did not seem to be important to obtain that information. DOPRI5 provided number of tolerance misses but it was not convenient to obtain the number of times deceived. And DIMEX5 did not conveniently provide a sufficiently accurate solution at tolerance 10^{-15} to determine the accuracy of individual steps as required in computing number of steps deceived at tolerance 10^{-12}. Also, "AM" designates the Adams-Moulton integrator of LSODE.

DIMEX2 Test Results

TABLE 22. Order 2 Problem A1 Test.

Tolerance	10^{-3}		10^{-6}		10^{-9}	
method	DIM2	AM2	DIM2	AM2	DIM2	AM2
# steps	121	107	1222	1049	12231	10484
# func calls	126	116	1227	1055	12235	10490
# tol misses	1	**	1	**	0	**
# deceived	0	0	0	**	0	**
end relerr	7.9e-2	1.3e-1	8.8e-4	6.8e-4	8.9e-6	6.1e-6

TABLE 23. Order 2 Problem A2 Test.

Tolerance	10^{-3}		10^{-6}		10^{-9}	
method	DIM2	AM2	DIM2	AM2	DIM2	AM2
# steps	27	29	239	217	2356	2075
# func calls	31	32	243	220	2360	2078
# tol misses	0	**	0	**	0	**
# deceived	0	**	0	**	0	**
end relerr	3.4e-3	4.3e-3	4.8e-5	3.5e-5	5.0e-7	3.3e-7

TABLE 24. Order 2 Problem A3 Test.

Tolerance	10^{-3}		10^{-6}		10^{-9}	
method	DIM2	AM2	DIM2	AM2	DIM2	AM2
# steps	131	108	1160	950	11485	9430
# func calls	171	132	1199	1002	11521	9523
# tol misses	33	**	31	**	32	**
# deceived	2	**	1	**	0	**
end relerr	1.3e-3	4.6e-3	4.2e-5	4.6e-5	5.8e-7	4.2e-7

TABLE 25. Order 2 Problem A4 Test.

Tolerance	10^{-3}		10^{-6}		10^{-9}	
method	DIM2	AM2	DIM2	AM2	DIM2	AM2
# steps	19	18	169	144	1650	1377
# func calls	25	26	178	156	1659	1386
# tol misses	2	**	5	**	5	**
# deceived	0	**	0	**	0	**
end relerr	2.5e-4	4.1e-4	1.6e-6	1.9e-7	7.0e-9	5.8e-9

TABLE 26. Order 2 Problem A5 Test.

Tolerance	10^{-3}		10^{-6}		10^{-9}	
method	DIM2	AM2	DIM2	AM2	DIM2	AM2
# steps	24	21	185	162	1801	1539
# func calls	37	33	197	179	1815	1566
# tol misses	9	**	8	**	10	**
# deceived	0	**	11	**	9	**
end relerr	1.5e-1	2.7e-2	1.6e-3	8.4e-4	1.6e-5	1.0e-5

TABLE 27. Order 2 Problem B1 Test.

Tolerance	10^{-3}		10^{-6}		10^{-9}	
method	DIM2	AM2	DIM2	AM2	DIM2	AM2
# steps	244	191	2194	1724	21944	17060
# func calls	290	224	2199	1766	21949	17102
# tol misses	42	**	1	**	1	**
# deceived	32	**	1	**	0	**
end relerr	4.9e-3	5.4e-1	8.6e-5	5.4e-4	1.1e-6	3.5e-6

TABLE 28. Order 2 Problem B2 Test.

Tolerance	10^{-3}		10^{-6}		10^{-9}	
method	DIM2	AM2	DIM2	AM2	DIM2	AM2
# steps	92	90	275	239	2423	1930
# func calls	101	151	284	276	2427	1936
# tol misses	5	**	5	**	0	**
# deceived	0	**	3	**	0	**
end relerr	1.2e-4	4.2e-5	2.4e-7	2.6e-8	2.5e-10	1.1e-9

TABLE 29. Order 2 Problem B3 Test.

Tolerance	10^{-3}		10^{-6}		10^{-9}	
method	DIM2	AM2	DIM2	AM2	DIM2	AM2
# steps	37	45	259	209	2548	2014
# func calls	45	72	263	212	2552	2017
# tol misses	4	**	0	**	0	**
# deceived	0	**	0	**	0	**
end relerr	4.5e-4	3.7e-4	1.7e-5	1.7e-5	1.9e-7	1.4e-7

TABLE 30. Order 2 Problem B4 Test.

Tolerance	10^{-3}		10^{-6}		10^{-9}	
method	DIM2	AM2	DIM2	AM2	DIM2	AM2
# steps	191	143	1882	1356	18833	13705
# func calls	197	150	1886	1363	18837	13713
# tol misses	2	**	0	**	0	**
# deceived	8	**	0	**	0	**
end relerr	1.9e-1	1.0e-1	1.8e-3	1.7e-3	1.8e-5	1.6e-5

TABLE 31. Order 2 Problem B5 Test.

Tolerance	10^{-3}		10^{-6}		10^{-9}	
method	DIM2	AM2	DIM2	AM2	DIM2	AM2
# steps	114	87	1048	788	10378	7686
# func calls	133	150	1066	821	10384	7724
# tol misses	15	**	14	**	3	**
# deceived	10	**	0	**	0	**
end relerr	1.6e-2	1.1e-1	3.7e-4	6.1e-4	3.1e-6	5.0e-6

TABLE 32. Order 2 Problem C1 Test.

Tolerance	10^{-3}		10^{-6}		10^{-9}	
method	DIM2	AM2	DIM2	AM2	DIM2	AM2
# steps	54	50	356	247	3523	2383
# func calls	60	80	360	250	3527	2386
# tol misses	2	**	0	**	0	**
# deceived	0	**	0	**	0	**
end relerr	2.1e-4	2.1e-4	1.3e-5	2.6e-5	1.4e-7	1.8e-7

TABLE 33. Order 2 Problem C2 Test.

Tolerance	10^{-3}		10^{-6}		10^{-9}	
method	DIM2	AM2	DIM2	AM2	DIM2	AM2
# steps	268	225	394	325	2961	1949
# func calls	276	406	403	439	2970	1995
# tol misses	4	**	5	**	5	**
# deceived	0	**	2	**	0	**
end relerr	1.5e-4	2.7e-5	3.9e-7	1.3e-7	3.5e-10	6.9e-10

TABLE 34. Order 2 Problem C3 Test.

Tolerance	10^{-3}		10^{-6}		10^{-9}	
method	DIM2	AM2	DIM2	AM2	DIM2	AM2
# steps	116	112	259	202	2352	1567
# func calls	123	205	266	245	2356	1570
# tol misses	3	**	3	**	0	**
# deceived	1	**	0	**	0	**
end relerr	5.5e-2	9.5e-3	1.9e-4	1.2e-4	4.5e-6	5.0e-6

TABLE 35. Order 2 Problem C4 Test.

Tolerance	10^{-3}		10^{-6}		10^{-9}	
method	DIM2	AM2	DIM2	AM2	DIM2	AM2
# steps	117	93	259	165	2355	1199
# func calls	124	167	266	199	2359	1202
# tol misses	3	**	3	**	0	**
# deceived	1	**	3	**	1	**
end relerr	1.4e-2	2.5e-2	1.8e-4	2.0e-4	4.1e-6	7.0e-6

TABLE 36. Order 2 Problem C5 Test.

Tolerance	10^{-3}		10^{-6}		10^{-9}	
method	DIM2	AM2	DIM2	AM2	DIM2	AM2
# steps	30	16	287	157	2865	1536
# func calls	34	21	291	159	2869	1538
# tol misses	0	**	0	**	0	**
# deceived	0	**	0	**	1	**
end relerr	1.4e-3	4.9e-3	1.2e-5	1.2e-5	1.1e-7	1.9e-7

TABLE 37. Order 2 Problem D1 Test.

Tolerance	10^{-3}		10^{-6}		10^{-9}	
method	DIM2	AM2	DIM2	AM2	DIM2	AM2
# steps	130	115	1255	937	12541	9418
# func calls	135	168	1259	944	12545	9426
# tol misses	1	**	0	**	1	**
# deceived	6	**	0	**	0	**
end relerr	7.6e-1	2.1e+0	4.0e-3	4.6e-3	3.4e-5	9.4e-6

TABLE 38. Order 2 Problem D2 Test.

Tolerance	10^{-3}		10^{-6}		10^{-9}	
method	DIM2	AM2	DIM2	AM2	DIM2	AM2
# steps	155	129	1483	1071	14816	10659
# func calls	165	199	1487	1091	14820	10680
# tol misses	6	**	0	**	0	**
# deceived	11	**	0	**	0	**
end relerr	6.3e-1	2.0e+0	5.7e-3	2.5e-3	5.8e-5	2.1e-5

TABLE 39. Order 2 Problem D3 Test.

Tolerance	10^{-3}		10^{-6}		10^{-9}	
method	DIM2	AM2	DIM2	AM2	DIM2	AM2
# steps	202	152	1799	1289	17937	12746
# func calls	236	230	1803	1322	17941	12779
# tol misses	30	**	0	**	0	**
# deceived	6	**	0	**	0	**
end relerr	7.1e-1	2.2e+0	7.6e-3	1.7e-3	8.2e-5	4.1e-5

TABLE 40. Order 2 Problem D4 Test.

Tolerance	10^{-3}		10^{-6}		10^{-9}	
method	DIM2	AM2	DIM2	AM2	DIM2	AM2
# steps	269	188	2294	1634	22896	16255
# func calls	330	256	2298	1683	22900	16306
# tol misses	57	**	0	**	0	**
# deceived	8	**	0	**	0	**
end relerr	7.0e-1	1.8e+0	8.9e-3	3.0e-3	9.3e-5	5.6e-5

TABLE 41. Order 2 Problem D5 Test.

Tolerance	10^{-3}		10^{-6}		10^{-9}	
method	DIM2	AM2	DIM2	AM2	DIM2	AM2
# steps	414	280	3457	2464	34502	24212
# func calls	521	363	3461	2545	34506	24296
# tol misses	103	**	0	**	0	**
# deceived	11	**	0	**	1	**
end relerr	5.4e-1	1.2e+0	9.3e-3	5.2e-3	9.3e-5	7.4e-5

TABLE 42. Order 2 Problem E1 Test.

Tolerance	10^{-3}		10^{-6}		10^{-9}	
method	DIM2	AM2	DIM2	AM2	DIM2	AM2
# steps	74	62	738	611	7372	6021
# func calls	78	117	742	614	7376	6024
# tol misses	0	**	0	**	0	**
# deceived	8	**	0	**	0	**
end relerr	2.7e-1	2.6e-1	2.0e-3	1.8e-3	2.0e-5	1.5e-5

TABLE 43. Order 2 Problem E2 Test.

Tolerance	10^{-3}		10^{-6}		10^{-9}	
method	DIM2	AM2	DIM2	AM2	DIM2	AM2
# steps	282	219	2649	2010	26490	20016
# func calls	318	264	2657	2061	26495	20071
# tol misses	32	**	4	**	1	**
# deceived	28	**	0	**	0	**
end relerr	2.2e-2	1.3e-2	1.9e-4	1.5e-4	1.9e-6	1.6e-6

TABLE 44. Order 2 Problem E3 Test.

Tolerance	10^{-3}		10^{-6}		10^{-9}	
method	DIM2	AM2	DIM2	AM2	DIM2	AM2
# steps	281	213	2714	2064	27128	20440
# func calls	304	230	2718	2086	27133	20463
# tol misses	19	**	0	**	1	**
# deceived	3	**	0	**	0	**
end relerr	9.7e-2	1.4e-1	1.2e-3	1.2e-3	1.1e-5	1.3e-5

TABLE 45. Order 2 Problem E4 Test.

5olerance	10^{-3}		10^{-6}		10^{-9}	
method	DIM2	AM2	DIM2	AM2	DIM2	AM2
# steps	16	14	138	104	1365	1052
# func calls	22	23	142	109	1369	1057
# tol misses	2	**	0	**	0	**
# deceived	1	**	5	**	0	**
end relerr	2.8e-4	3.7e-4	4.3e-6	3.7e-6	4.3e-8	3.8e-8

TABLE 46. Order 2 Problem E5 Test.

Tolerance	10^{-3}		10^{-6}		10^{-9}	
method	DIM2	AM2	DIM2	AM2	DIM2	AM2
# steps	31	28	280	209	2804	2047
# func calls	32	52	285	220	2809	2059
# tol misses	1	**	1	**	1	**
# deceived	23	**	0	**	0	**
end relerr	4.9e-3	3.5e-3	4.7e-5	4.2e-5	4.7e-7	4.4e-7

DIMEX5 TEST RESULTS

TABLE 47. Order 5 Problem A1 Test.

Tolerance	10^{-6}		10^{-9}		10^{-12}	
method	DIM5	DOPRI5	DIM5	DOPRI5	DIM5	DOPRI5
# steps	97	94	288	361	882	1426
# func calls	531	566	1496	2168	4436	8558
# tol misses	5	0	7	0	1	0
# deceived	0	**	0	**	0	**
end relerr	1.1e-5	3.9e-6	5.0e-8	3.3e-9	1.7e-10	3.1e-12

TABLE 48. Order 5 Problem A2 Test.

Tolerance	10^{-6}		10^{-9}		10^{-12}	
method	DIM5	DOPRI5	DIM5	DOPRI5	DIM5	DOPRI5
# steps	38	22	104	67	320	246
# func calls	226	134	546	404	1621	1478
# tol misses	3	0	1	0	0	0
# deceived	0	**	0	**	0	**
end relerr	8.7e-7	4.4e-7	5.1e-9	2.8e-10	1.7e-11	1.9e-13

TABLE 49. Order 5 Problem A3 Test.

Tolerance	10^{-6}		10^{-9}		10^{-12}	
method	DIM5	DOPRI5	DIM5	DOPRI5	DIM5	DOPRI5
# steps	136	88	426	286	1348	1070
# func calls	721	530	2166	1718	6776	6422
# tol misses	4	15	3	19	3	19
# deceived	0	**	0	**	0	**
end relerr	4.3e-6	2.1e-6	8.8e-9	1.7e-9	2.6e-11	2.1e-12

TABLE 50. Order 5 Problem A4 Test.

Tolerance	10^{-6}		10^{-9}		10^{-12}	
method	DIM5	DOPRI5	DIM5	DOPRI5	DIM5	DOPRI5
# steps	23	18	64	56	197	205
# func calls	146	110	346	338	1016	1232
# tol misses	2	1	1	2	2	1
# deceived	0	**	0	**	0	**
end relerr	1.9e-7	7.2e-8	3.2e-10	5.1e-11	9.8e-14	6.9e-13

TABLE 51. Order 5 Problem A5 Test.

Tolerance	10^{-6}		10^{-9}		10^{-12}	
method	DIM5	DOPRI5	DIM5	DOPRI5	DIM5	DOPRI5
# steps	28	15	78	51	241	190
# func calls	171	92	411	308	1226	1142
# tol misses	2	0	0	1	0	1
# deceived	2	**	1	**	**	**
end relerr	5.4e-5	6.5e-6	2.3e-7	7.6e-9	3.5e-10	8.9e-12

TABLE 52. Order 5 Problem B1 Test.

Tolerance	10^{-6}		10^{-9}		10^{-12}	
method	DIM5	DOPRI5	DIM5	DOPRI5	DIM5	DOPRI5
# steps	233	165	683	583	2147	2311
# func calls	1201	992	3436	3500	10756	13868
# tol misses	3	17	0	3	0	0
# deceived	2	**	1	**	**	**
end relerr	2.1e-4	1.9e-5	9.9e-7	2.5e-9	3.3e-9	2.3e-12

TABLE 53. Order 5 Problem B2 Test.

Tol (abs)	10^{-6}		10^{-9}		10^{-12}	
method	DIM5	DOPRI5	DIM5	DOPRI5	DIM5	DOPRI5
# steps	87	43	134	123	354	450
# func calls	486	260	711	740	1806	2702
# tol misses	6	2	4	0	3	0
# deceived	6	**	0	**	**	**
end relerr	3.6e-7	4.7e-7	2.9e-10	6.3e-11	2.1e-13	9.7e-14

TABLE 54. Order 5 Problem B3 Test.

Tol (abs)	10^{-6}		10^{-9}		10^{-12}	
method	DIM5	DOPRI5	DIM5	DOPRI5	DIM5	DOPRI5
# steps	49	33	112	108	344	400
# func calls	266	200	596	650	1741	2402
# tol misses	4	0	3	0	0	**
# deceived	1	**	0	**	**	**
end relerr	2.7e-7	9.7e-8	2.1e-9	1.9e-10	7.4e-12	1.4e-13

TABLE 55. Order 5 Problem B4 Test.

Tol (abs)	10^{-6}		10^{-9}		10^{-12}	
method	DIM5	DOPRI5	DIM5	DOPRI5	DIM5	DOPRI5
# steps	159	108	494	424	1561	1682
# func calls	816	650	2491	2546	7826	10094
# tol misses	0	0	0	0	0	0
# deceived	0	**	0	**	**	**
end relerr	3.1e-6	3.4e-6	5.8e-9	8.1e-10	2.8e-11	8.9e-13

TABLE 56. Order 5 Problem B5 Test.

Tol (abs)	10^{-6}		10^{-9}		10^{-12}	
method	DIM5	DOPRI5	DIM5	DOPRI5	DIM5	DOPRI5
# steps	137	82	422	284	1327	1123
# func calls	706	494	2136	1706	6661	6740
# tol misses	0	7	1	**	1	0
# deceived	0	**	0	**	**	**
end relerr	6.8e-6	3.0e-6	3.2e-8	7.7e-9	8.9e-11	9.0e-12

TABLE 57. Order 5 Problem C1 Test.

Tol (abs)	10^{-6}		10^{-9}		10^{-12}	
method	DIM5	DOPRI5	DIM5	DOPRI5	DIM5	DOPRI5
# steps	51	39	132	134	411	508
# func calls	281	236	681	806	2076	3050
# tol misses	1	0	0	0	0	0
# deceived	9	**	9	**	**	**
end relerr	1.1e-7	1.7e-7	1.5e-9	4.8e-10	5.7e-12	5.5e-13

TABLE 58. Order 5 Problem C2 Test.

Tol (abs)	10^{-6}		10^{-9}		10^{-12}	
method	DIM5	DOPRI5	DIM5	DOPRI5	DIM5	DOPRI5
# steps	198	68	214	142	466	496
# func calls	1086	410	1126	854	2381	2978
# tol misses	15	2	7	2	6	0
# deceived	17	**	43	**	**	**
end relerr	2.6e-8	7.2e-7	7.8e-11	2.2e-10	2.2e-13	2.7e-13

TABLE 59. Order 5 Problem C3 Test.

Tol (abs)	10^{-6}		10^{-9}		10^{-12}	
method	DIM5	DOPRI5	DIM5	DOPRI5	DIM5	DOPRI5
# steps	98	41	122	108	334	398
# func calls	536	248	646	650	1691	2390
# tol misses	5	1	3	0	0	0
# deceived	0	**	0	**	**	**
end relerr	1.4e-5	3.2e-5	2.2e-8	1.4e-8	1.9e-10	1.8e-11

TABLE 60. Order 5 Problem C4 Test.

Tol (abs)	10^{-6}		10^{-9}		10^{-12}	
method	DIM5	DOPRI5	DIM5	DOPRI5	DIM5	DOPRI5
# steps	104	39	132	94	334	340
# func calls	561	236	696	566	1691	2042
# tol misses	4	1	3	0	0	0
# deceived	0	**	0	**	**	**
end relerr	7.0e-7	3.5e-5	7.6e-9	3.0e-8	1.5e-10	3.4e-11

TABLE 61. Order 5 Problem C5 Test.

Tolerance	10^{-6}		10^{-9}		10^{-12}	
method	DIM5	DOPRI5	DIM5	DOPRI5	DIM5	DOPRI5
# steps	14	15	32	55	94	211
# func calls	91	92	181	332	491	1268
# tol misses	0	0	0	2	0	4
# deceived	0	**	0	**	**	**
end relerr	1.8e-6	1.8e-6	6.7e-9	8.3e-10	1.8e-11	5.7e-13

TABLE 62. Order 5 Problem D1 Test.

Tol (abs)	10^{-6}		10^{-9}		10^{-12}	
method	DIM5	DOPRI5	DIM5	DOPRI5	DIM5	DOPRI5
# steps	119	83	373	323	1177	1276
# func calls	616	500	1891	1940	5911	7658
# tol misses	0	0	1	0	1	0
# deceived	0	**	1	**	**	**
end relerr	1.3e-4	3.1e-4	8.4e-7	1.3e-7	2.5e-9	1.7e-10

TABLE 63. Order 5 Problem D2 Test.

Tol (abs)	10^{-6}		10^{-9}		10^{-12}	
method	DIM5	DOPRI5	DIM5	DOPRI5	DIM5	DOPRI5
# steps	163	98	512	350	1617	1382
# func calls	836	590	2581	2102	8106	8294
# tol misses	0	6	0	0	0	0
# deceived	0	**	0	**	**	**
end relerr	1.5e-4	4.3e-4	4.5e-7	1.2e-7	1.4e-9	1.5e-9

TABLE 64. Order 5 Problem D3 Test.

Tol (abs)	10^{-6}		10^{-9}		10^{-12}	
method	DIM5	DOPRI5	DIM5	DOPRI5	DIM5	DOPRI5
# steps	212	131	671	414	2121	1638
# func calls	1081	788	3376	2486	10631	9830
# tol misses	0	20	0	0	1	0
# deceived	0	**	0	**	**	**
end relerr	4.0e-4	2.6e-4	1.0e-6	9.1e-8	2.8e-9	1.1e-10

TABLE 65. Order 5 Problem D4 Test.

Tol (abs)	10^{-6}		10^{-9}		10^{-12}	
method	DIM5	DOPRI5	DIM5	DOPRI5	DIM5	DOPRI5
# steps	281	174	889	514	2807	2035
# func calls	1426	1046	4471	3086	14061	12212
# tol misses	0	34	1	0	1	0
# deceived	0	**	0	**	**	**
end relerr	6.8e-4	2.0e-4	1.5e-6	8.0e-8	4.3e-9	9.7e-11

TABLE 66. Order 5 Problem D5 Test.

Tol (abs)	10^{-6}		10^{-9}		10^{-12}	
method	DIM5	DOPRI5	DIM5	DOPRI5	DIM5	DOPRI5
# steps	422	256	1331	728	4201	2884
# func calls	2136	1538	6686	4370	21056	17306
# tol misses	1	54	0	0	0	0
# deceived	1	**	1	**	**	**
end relerr	67.1-5	1.3e-4	9.1e-8	6.0e-8	2.2e-9	6.7e-11

TABLE 67. Order 5 Problem E1 Test.

Tolerance	10^{-6}		10^{-9}		10^{-12}	
method	DIM5	DOPRI5	DIM5	DOPRI5	DIM5	DOPRI5
# steps	94	112	286	426	900	1683
# func calls	491	674	1451	2558	4521	10100
# tol misses	0	10	0	11	0	12
# deceived	0	**	0	**	**	**
end relerr	1.1e-5	2.2e-6	4.3e-8	2.7e-9	1.3e-10	2.8e-12

TABLE 68. Order 5 Problem E2 Test.

Tol (abs)	10^{-6}		10^{-9}		10^{-12}	
method	DIM5	DOPRI5	DIM5	DOPRI5	DIM5	DOPRI5
# steps	303	183	916	637	2922	2503
# func calls	1551	1100	4601	3824	14631	15020
# tol misses	3	18	0	6	0	5
# deceived	2	**	1	**	**	**
end relerr	5.0e-6	2.0e-4	4.3e-9	6.1e-8	7.3e-12	2.6e-11

TABLE 69. Order 5 Problem E3 Test.

Tol (abs)	10^{-6}		10^{-9}		10^{-12}	
method	DIM5	DOPRI5	DIM5	DOPRI5	DIM5	DOPRI5
# steps	220	137	695	489	2198	1916
# func calls	1121	824	3501	2936	11011	11498
# tol misses	0	10	1	4	0	0
# deceived	0	**	0	**	**	**
end relerr	8.8e-6	1.6e-4	4.4e-8	4.9e-8	4.0e-10	2.3e-11

TABLE 70. Order 5 Problem E4 Test.

Tol (abs)	10^{-6}		10^{-9}		10^{-12}	
method	DIM5	DOPRI5	DIM5	DOPRI5	DIM5	DOPRI5
# steps	19	17	47	46	145	167
# func calls	121	104	256	278	746	1004
# tol misses	1	1	0	0	0	0
# deceived	0	**	0	**	**	**
end relerr	3.9e-8	1.8e-7	1.3e-10	1.5e-10	1.3e-13	1.7e-13

TABLE 71. Order 5 Problem E5 Test.

Tol (abs)	10^{-6}		10^{-9}		10^{-12}	
method	DIM5	DOPRI5	DIM5	DOPRI5	DIM5	DOPRI5
# steps	25	24	78	61	242	233
# func calls	146	146	411	368	1231	1400
# tol misses	0	5	0	0	0	0
# deceived	0	**	0	**	**	**
end relerr	1.3e-6	1.6e-8	4.3e-9	1.5e-11	1.2e-11	1.2e-14

DISCUSSION OF TEST RESULTS

In scanning the results for second order testing, it is evident that the number of steps is usually 10 to 20% higher for the DIMEX2 integrator in comparison with LSODE. But since LSODE has been extensively refined over a period of nearly 20 years, this is not bad for a newly developed research code. A different DIMSIM with a smaller error constant would produce fewer steps with otherwise the same implementation. On the other hand, for these relatively easy problems, LSODE minimizes the number of function iterations to the extent that often the average number of function evaluations required is just over 1 per step. DIMEX2 significantly outperforms LSODE at loose tolerances on problems A2, A4, B2, B3, B5, C1, C2, C3,C4, E1, E4, and E5. The orbital problems D1 and D2 might also be included, but the accuracy of both solvers was inadequate to make these true tests. The operation of DIMEX2 shows that although it is not really a competitive method for simple problems with tighter tolerances, the implementation is essentially correct and appropriate. And it points to the possibility of developing a truly competitive DIMSIM second order explicit solver, since the number of function evaluations per step is consistently 1 even at looser tolerances.

A calculation of the stability region for the FASAL implementation was performed. The following graph shows that the portion of the negative real axis included was reduced by 1/3. However, it was found that method parameter could be chosen to find another Type 1 second order DIMSIM with a more favorable stability region. The longest portion of the negative real axis that could be obtained is also shown and is unreduced from the original DIMSIM, although the portion off the real axis is seriously reduced for this case in Figure 14. It is felt that more can be done to produce desirable stability regions for higher orders with more parameters.

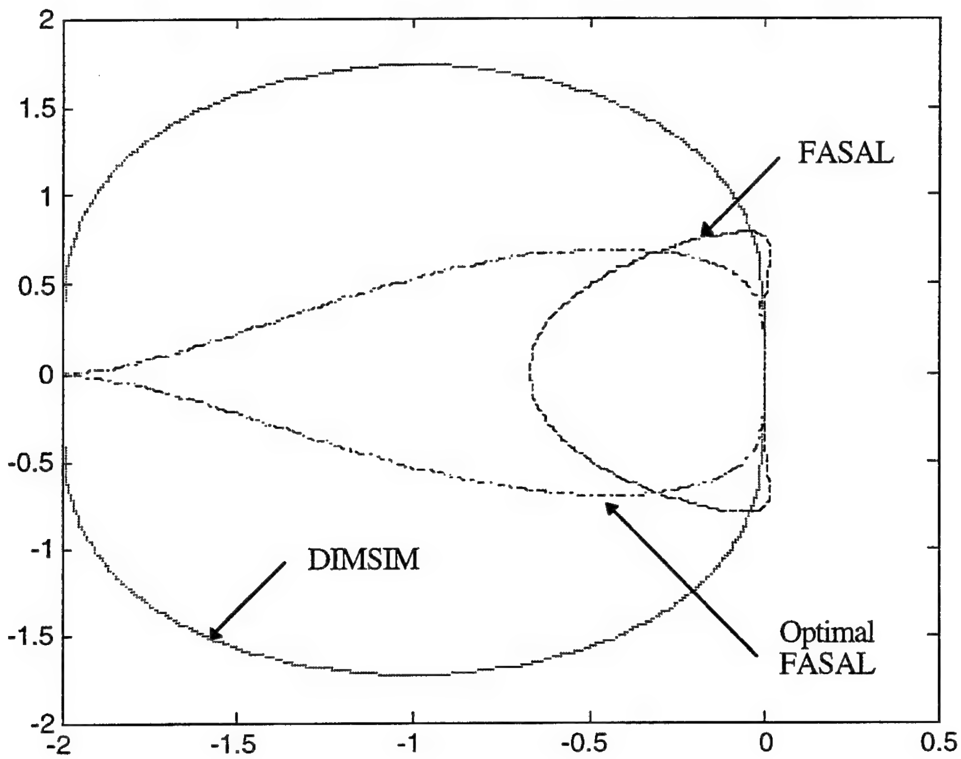


FIGURE 14. Comparison of Stability Regions for Second Order.

The fifth order results show very competitive results. DIMEX5 often beats DOPRI5, and otherwise provides roughly comparable results, and this despite the larger error constant (1/720 versus 1/3600, a factor of 5) and a slightly smaller stability region. This is directly attributable to the savings of one function evaluation per step with the DIMSIM. But this does not show the complete picture, because dense output is typically required in real applications, not just a race to the end of the interval. And the DIMSIM requires no additional function evaluations or system solves to provide interpolated output at 5th order, while DOPRI5 requires two additional stages to provide 5th order dense output (Reference 18). Thus when dense output requirements are considered, DIMEX5 in reality significantly outperforms DOPRI5 for much of this test set and is rarely beaten. And improved DIMSIMs may be found with smaller error constants and larger stability regions. Finally, the difference at higher orders is expected to be much greater. For example, at 8th order, the Runge-Kutta-based DOP853 code provides an 8th order method with an interpolant of 7th order with an effective number of 15 function evaluations per step (Reference 18), while an 8th order DIMSIM requires only 8 function evaluations per step and provides an interpolant of 8th order.

A note should be provided concerning FASAL operation. The reported tests were first conducted using the FASAL property only when step size had not changed since the last step, and for most problems the number of function evaluations was reduced by the

expected 20%. An attempt was made to allow this for all steps after the first, and in most problems the results were satisfactory and the number of function evaluations significantly reduced. However, for perhaps 5 or 6 of the problems the quality of the error estimate deteriorated significantly, resulting in unacceptably large numbers of times the error estimator was deceived, reducing accuracy, and causing up to one third more function evaluations. Further examination showed that for these problems, not using FASAL produced the best results. Table 72 shows what happened for Problem C2 with tolerance 10^{-9}.

TABLE 72. Order 5 With and Without FASAL for Problem C2 Test.

Use of FASAL	None	Constant Step	All After First
#Steps	234	357	590
#func calls	1286	1633	2386
#tol misses	19	7	1
#deceived	81	209	430
end relerr	3.0e-10	2.5e-11	2.0e-10

It was originally thought that this is due to larger higher order terms that are present in these problems, and this may be an important factor. However, a calculation showed that the stability region for order 5 became extremely limited with FASAL implementation (see Figure 15).

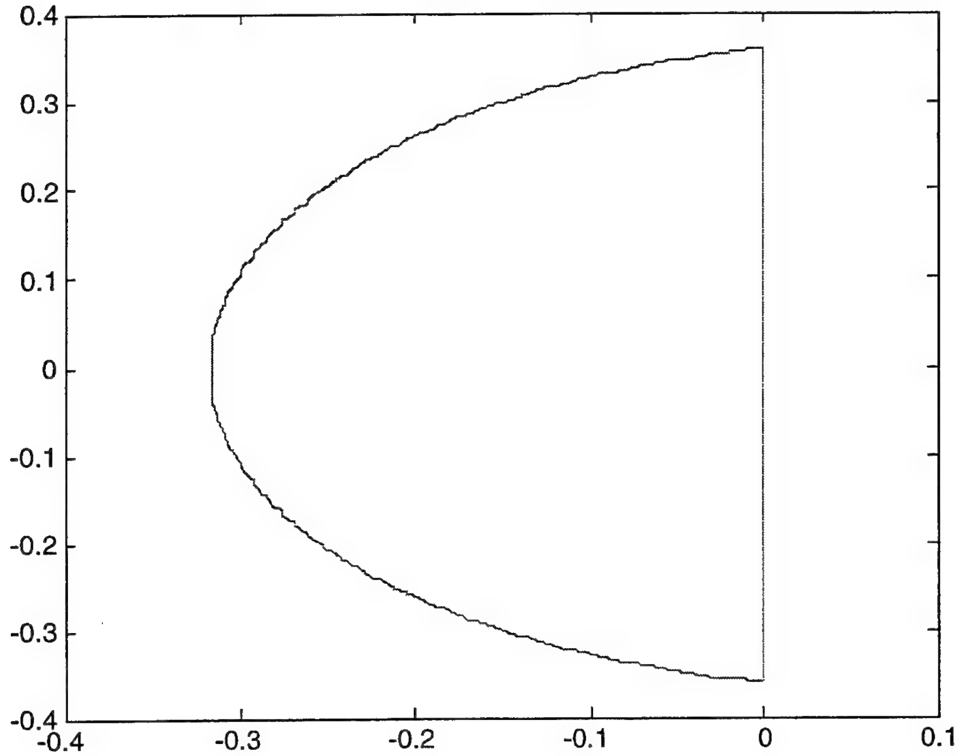


FIGURE 15. Order 5 FASAL Implementation Stability Region.

In Figure 15 z values are shown in a neighborhood of the origin where the eigenvalue of largest modulus w of the stability matrix has $|w| = 1$. The dimensions of this region are smaller by an order of magnitude as compared with the original DIMSIM. It is evident that stability for the FASAL implementation must be considered when deriving DIMSIMs if this implementation efficiency is to be utilized! As a result of the small stability region, a FASAL implementation was not used in the testing reported above.

For the DIMSIM itself, the error constant for the method is rather small, and at times the sixth order term might be dominated by the seventh order terms. If this is true, a different method might not have this problem. More study of the estimation of higher order terms and of the use of FASAL is needed.

FUTURE DEVELOPMENT

It might be noted that DIMEX5 frequently generates exactly 1 tolerance miss. This is usually generated on the second step, when the step size calculated for the first step is used for the second step, which has a larger associated error. A future version of the code will incorporate an appropriate modification similar to the approach used in DIMEX2. Other

improvements and optimizations are anticipated for later development and suggestions and bug reports are requested. It should also be noted that DIMEX5 uses one example of a fifth order Type 1 DIMSIM. It is expected that in the future other methods of this class will be found with different error constants and relatively smaller coefficients for higher order terms, and these may enable significant performance improvements. Stability for FASAL implementations will also be utilized in deriving future methods. And although Runge-Kutta stability regions are a design choice here, they are not necessarily essential and stability regions even larger than those of Runge-Kutta methods may be found. And the family, which now includes only orders 2 and 5, may already be extended to include methods through order 8. It is hoped that in time a single variable order code may be developed incorporating the entire family in an optimal way.

REFERENCES

1. J. C. Butcher. "Diagonally Implicit Multi-Stage Integration Methods," *Appl. Numer. Math.*, Vol. 11 (1993), pp. 347-63.

2. J. C. Butcher and Z. Jackiewicz. "Diagonally Implicit General Linear Methods for Ordinary Differential Equations," *BIT*, Vol. 33 (1993), pp. 452-472.

3. J. C. Butcher and Z. Jackiewicz. "Implementation of Diagonally Implicit Multistage Integration Methods for Ordinary Differential Equations," to appear in *SIAM J. Num. Anal.*

4. Z. Jackiewicz, R. Vermiglio, and M. Zennaro. "Variable Stepsize Diagonally Implicit Multistage Integration Methods for Ordinary Differential Equations," *Appl. Numer. Math.*, Vol. 16 (1995), pp. 343-67.

5. Z. Jackiewicz, R. Vermiglio, and M. Zennaro. "Regularity Properties of Multistage Integration Methods," submitted to *J. Comp. Appl. Math.*

6. J. C. Butcher, Z. Jackiewicz, and H. Mittelmann. "A Nonlinear Optimization Approach to the Construction of General Linear Methods of High Order," to appear in *J. Comp. Appl. Math.*

7. J. C. Butcher and Z. Jackiewicz. "Construction of Diagonally Implicit General Linear Methods of Type 1 and 2 for Ordinary Differential Equations," to appear in *Appl. Num. Math.*

8. J. C. Butcher and Z. Jackiewicz. "Construction of High Order DIMSIMs for Ordinary Differential Equations," submitted to *IMA J. Num. Anal.*

9. J. C. Butcher. "General Linear Methods," *Comp. Math. Appl. Int. J.* 31 (1996), 105-112.

10. K. Burrage, J. C. Butcher, and F. H. Chipman. "An Implementation of Singly-Implicit Runge-Kutta Methods," *BIT*, Vol. 20 (1980), pp. 326-40.

11. J. C. Butcher. *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods*. Chichester, Wiley and Sons, 1987.

12. J. D. Lambert. *Computational Methods in Ordinary Differential Equations*. London, Wiley and Sons, 1973.

13. C. W. Gear. "The Potential for Parallelism in Ordinary Differential Equations," University of Illinois Report No. UIUCDCS-R-86-1246, 1986.

14. K. Burrage. *Parallel and Sequential Methods for Ordinary Differential Equations*. Oxford, Clarendon Press, 1995.

15. S. Vandewalle. *Parallel Multigrid Waveform Relaxation for Parabolic Problems. Stuttgart*: B.G. Teubner, 1993.

16. J. Donelson and E. Hansen. "Cyclic Composite Multistep Predictor-Corrector Methods," *SIAM J. of Numer. Anal.*, Vol. 8 (March 1971), pp. 137-57.

17. A. Nordsieck. "On Numerical Integration of Ordinary Differential Equations," *Math. Comp.*, Vol. 15 (1962), pp. 22-49.

18. E. Hairer, S. P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations, Vol. I: Nonstiff Problems*, 2nd ed. New York, Springer-Verlag, 1993.

19. S. Trocogna. Implementation of Two-Step Runge-Kutta Methods for Ordinary Differential Equations," to appear in *J. Comp. Appl. Math.*

20. Z. Jackiewicz and S. Trocogna. "A General Class of Two-Step Runge-Kutta Methods for Ordinary Differential Equations," *SIAM J. Numer. Anal.* 32 (1995), pp. 1390-1427.

21. R. Enenkel. "DIMSEMs-Diagonally Implicit Single-Eigenvalue Methods for the Numerical Solution of Stiff Ordinary Differential Equations on Parallel Computers." Ph.D. dissertation, University of Toronto, 1996.

22. P. Albrecht, "Numerical Treatment of O.D.E.s: The Theory of A-Methods," *Numer. Math.*, Vol. 47 (1985), pp. 59-87.

23. Z. Jackiewicz, R. Vermiglio, and M. Zennaro. "Variable Stepsize Diagonally Implicit Multistage Integration Methods for Ordinary Differential Equations," *Appl. Numer. Math.*, Vol. 16 (1995), pp. 343-67.

24. L. F. Shampine and M. K. Gordon. *Computer Solution of Ordinary Differential Equations: The Inital Value Problem*. San Francisco, W. H. Freeman and Company, 1975.

25. I. Gladwell, L. F. Shampine, and R. W. Brankin. "Automatic Selection of the Initial Step Size for an ODE Solver," *J. Comp. Appl. Math.*, Vol. 18 (1987), pp. 175-92.

26. J. R. Dormand and P. J. Prince. "A Family of Embedded Runge-Kutta Formulae." *J. Comp. Appl. Math.*, Vol. 6, pp. 19-26.

27. A. Prothero and A. Robinson. "On the Stability and Accuracy of One-Step Methods for Solving Stiff Systems of Ordinary Differential Equations," *Math. Comp.*, Vol. 28 (1974), pp. 145-62.

28. Z. Jackiewicz and M. Zennaro. "Variable Step-Size Explicit Two-Step Runge-Kutta Methods," *Math. Comp.*, Vol. 59 (1992), pp. 421-438.

29. A. Bellen, Z. Jackiewicz, and M. Zennaro. "Local Error Estimation for Singly Implicit Formulas by Two-Step Runge-Kutta Methods," *BIT*, Vol. 32 (1992), pp. 104-17.

30. R. Enenkel and K. Jackson, "DIMSEMs-Diagonally Implicit Single-Eigenvalue Methods for the Numerical Solution of Stiff ODEs on Parallel Computers," to appear in *Adv. Comp. Math.*

31. T. E. Hull, W. H. Enright, B. M. Fellen, and A. E. Sedgwick. "Comparing Numerical Methods for Ordinary Differential Equations," *SIAM J. Numer. Anal.*, Vol. 9 (December 1972), pp. 603-637.

32. K. Radhakrishnan and A. Hindmarsh. "Description and Use of LSODE, the Livermore Solver for Ordinary Differential Equations," NASA Reference Publication 1327, Lawrence Livermore National Laboratory Report UCRL-ID-113855, December 1993.

Appendix

DIMEX5 Code

```
subroutine dim51(X0,Y0,X,Y,F,NEqn,XF,H,ATol,RTol,Starting,XOut,
   1         NCalls,NMisses,Diag,ErrEst,Work)
c*********************************************************************
****
c
c   dim51 calls driver which controls the integration process. Main
c   function of dim51 is to divide up work array so its use is transparent
c   to user.  Work array is dimensioned at least 52*NEqn+2.
c  Parameter definitions:
c  X0:  (real*8) Initial value of independent variable at start of integration,
c           input
c  Y0:  (real*8 array) Initial value of solution at start of integration,
c           dimensioned NEqn, input
c  X:  (real*8) Desired output point if Diag is .false., otherwise set to same
c           as XF, input
c  Y:  (real*8) Computed solution at X if Diag is .false., otherwise at XOut,
c           output only
c  F:  (subroutine) Provides derivative function f for ODE y'=f(x,y).
c           Subroutine must be defined with parameters F(X,Y,YP,NEqn),
c           where YP is the derivative at (X,Y) and NEqn is as defined
c           below.  F must be declared external in the calling program.
c  NEqn:  (integer) Number of equations in ODE system, input
c  XF: (real*8) Intended termination point for integration process, input
c  H:  (real*8) Step size used.  Output only, not to be defined or changed by
c          the user
c  ATol: (real*8) Absolute tolerance for local truncation error, used in
c          controlling integration according to formula described below,
c          input
c  RTol:  (real*8) Relative tolerance for local truncation error, used in
c          controlling integration according to formula
c          ErrEst<ATol+RTol* , input
c  Starting:  (logical) Initially set to .true., thereafter not to be changed
c          by the user
c  XOut::  (real*8) End point of last integration step performed,output only,
c          not to be changed by the user
c  NCalls:  (integer) Number of function calls since beginning of integration.
c          Output only, not to be changed by the user.
c  NMisses:  (integer) Number of tolerance misses since beginning of
c          integration.  Output only, not to be changed by the user.
```

```
c Diag:  (logical) Set to true if only one integration step at a time is
c          desired, which is the preferred mode for study of integrator
c          behavior, and false if it is desired to integrate until
c          interpolated output at X may be obtained
c ErrEst:  (real*8) Maximum norm of estimated error in last integration step
c          taken, output only
c Work:  (real*8) Work array which must be dimensioned at least 2+23*NEqn
c          for 2nd order  and 2+52*NEqn for 5th order.
c
c***********************************************************************
****
      implicit none
      logical Starting,Diag
      integer NCalls,NMisses,NEqn
      real*8 H,X0,Y0(NEqn),X,Y(NEqn),ATol,RTol,XF,XOut,Work(2+52*NEqn)
      real*8 ErrEst
      external F
      call driver(X0,Y0,X,Y,F,NEqn,XF,H,ATol,RTol,Starting,XOut,
     1          NCalls,NMisses,Diag,ErrEst,Work(1),Work(2),
     2          Work(2+5*NEqn),Work(2+10*NEqn),Work(2+15*NEqn),
     3          Work(3+15*NEqn),
     4          Work(3+16*NEqn),Work(3+17*NEqn),Work(3+18*NEqn),
     5          Work(3+19*NEqn),Work(3+20*NEqn),Work(3+21*NEqn),
     6          Work(3+22*NEqn),Work(3+27*NEqn),Work(3+32*NEqn),
     7          Work(3+37*NEqn),Work(3+38*NEqn),Work(3+46*NEqn))
      end
```

```
subroutine driver(X0,Y0,X,Y,F,NEqn,XF,H,ATol,RTol,Starting,XOut,
   1         NCalls,NMisses,Diag,ErrEst,HOld,YIter,FStage,YIterP,
   2         XOld,YP,Y1,Y2,Y4,Y1P,Y2P,YStage,
   3         YIterN,YIterS,FStageN,Y6Der,RKWork,NordVec)
c*********************************************************************
****
c driver provides the essential logic to control the integration process and
c calls the subprograms needed to carry out the necessary calculations.  When
c Starting is set to .true., start is called to obtain an initial Nordsieck
c vector, step size and external stage vector.  solver then is called to
c carry out the first integration step and the error estimate is checked to
c ensure the result is within tolerance.  If not the step size is halved and
c Nordsieck vector is rescaled to produce a new starting external stage vector.
c This process is repeated until successful.  Starting is then set to .false.
c and the integration process proceeds as NewStep calculates a new step size to
c use.  If Diag is .true. driver returns dim51 which returns to be called again
c by the main program, otherwise at each iteration either XOut is at least X,
c in which case NordInterp produces an interpolated solution value at X and a
c return is made to the main program for another call, or an integration step
c is carried out.  An integration step begins by saving copies of the current
c and previous external stage vectors.  Then rescale is called to rescale the
c current external stage vector, a DIMSIM step is performed using a call to
c solver, and ErrEst is checked to be sure the error is within tolerance.  If
c not, a tolerance miss is counted and the step size is halved.  Rescaling
c again takes place using the saved values, and this process continues until
c the step is within tolerance or MaxTries is exceeded, generating an error
c message and STOP.  For normal execution final XOut is XF.
c
c  Parameters:
c  X0,Y0,X,Y,F,NEqn,XF,H,ATol,RTol,Starting,XOut,NCalls,NMisses,Diag,ErrEst:
c       As in dim51, see Section a above.
c HOld:  (real*8) Step size used for previous successful integration step
c YIter:  (real*8 array) External stage vector for current step, dimensioned
c       (NEqn,5)
c FStage:  (real*8 array) Vector of derivative values computed from internal
c       stage values for most recent successful step, dimensioned (NEqn,5).
c YIterP: (real*8 array) External stage vector for previous successful step,
c       dimensioned (NEqn,5)
c XOld:  (real*8) Previous successful step point
c YP:  (real*8 array) Derivative at X0, =f(X0,Y0).  Dimensioned NEqn.
c Y1,Y2,Y4,Y1P,Y2P, Y6Der:  (real*8 arrays) Storage needed for subroutine start.
c      Each dimensioned NEqn.
c YIterN,YIterS, FStageN:  (real*8 arrays) Storage needed for temporary storage
c      of current and previous external stage vectors and current vector of
c      derivatives at internal stage points until it is verified that step
c      was successful, dimensioned (NEqn,5).
c YStage:  (real*8 array) Storage needed for subroutine solver, dimensioned
c      NEqn.
c RKWork: (real*8 array) Storage needed for subroutine start, dimensioned
c      8*NEqn.
c NordVec: (real*8 array) At first step, Nordsieck vector at X0.  Subsequently,
```

```
c      Nordsieck vector at XOut.  Dimensioned (NEqn,6)
c**********************************************************************
***
c
c   Driver controls the integration process.
c
      implicit none
      logical Starting,Diag
       real*8 H,X0,Y0(NEqn),ErrEst,FStage(NEqn,5),YIter(NEqn,5),X,XOut
      real*8 YIterN(NEqn,5),YIterP(NEqn,5),Y(NEqn),XOld,HOld,HFac,delta
      real*8 Theta,Eps,YIterS(NEqn,5),HNew,YP(NEqn),ATol,RTol,Y1P(NEqn)
      real*8 FStageN(NEqn,5),Y1(NEqn),YStage(NEqn)
      real*8 NewStep,X1,norm,Tol,Y2(NEqn),Y6Der(NEqn),NordVec(NEqn,6),XF
       real*8 C(5),A(5,5),B(5,5),vT(5),W(5,6),BT(6,5),Beta(5),Gamma(5)
      real*8 RKWork(8*NEqn),Y2P(NEqn),Y4(NEqn)
      common /Method/C,A,B,vT,W,BT,Beta,Gamma
      save Method
      integer NCalls,NMisses,iii,jjj,NEqn,JEqn,NTries,MaxTries
      external F
      parameter (Eps=2.2D-16,MaxTries=10)
      if (Starting) then
        NCalls=0
        NMisses=0
        Tol=ATol+RTol*norm(Y0,NEqn)
c
c   Compute starting values
c
        call start(YIter,F,X0,Y0,NEqn,H,YP,Tol,Y1,Y1P,Y6Der,Y2,
     1             Y2P,Y4,NordVec,RKWork)
        HOld=H
        NCalls=NCalls+21
        delta=1
c
c   Iterate until H yields error estimate within tolerance
c
1       continue
c
c   Compute First Step with Error Estimate and Interp Parameters
c
          call Solver(X0,NEqn,YIter,H,HOld,F,ErrEst,FStage,YIterP,
     1             Starting,YStage)
          NCalls=NCalls+5
        if (ErrEst .gt. Tol) then
            print *,'FIRST STEP MISSED!!! H',H,'X0=',X0
          NMisses=NMisses+1
          H=H/2
          delta=delta/2
          do JEqn=1,NEqn
            YIter(JEqn,1)=Y0(JEqn)
            do iii=2,5
              YIter(JEqn,iii)=Y0(JEqn)
```

```
            do jjj=2,6
              YIter(JEqn,iii)=YIter(JEqn,iii)+
     1          W(iii,jjj)*NordVec(JEqn,jjj)*delta**(jjj-1)
            end do
          end do
        end do
        goto 1
      endif
c      HNew=NewStep(H,Tol,2*ErrEst)
      XOut=X0+H
      HOld=H
c      H=HNew
      Starting = .false.
      if (Diag) then
        do JEqn=1,NEqn
          Y(JEqn)=YIter(JEqn,1)
        end do
        RETURN
      endif
    endif
c
c   Iterate until X is reached or exceeded.
c
2   continue
      if ((XOut .ge. X) .and. .not. Diag) then
c
c   Calculate Y using NordInterp
c
        Theta=(X-XOld)/HOld
        call NordInterp(Y,HOld,Theta,FStage,YIterP,NordVec,NEqn)
      else
        NTries=0
        XOld=XOut
        Tol=ATol+RTol*norm(YIter(1,1),NEqn)
c
c   Iterate until error estimate is within tolerance.
c
3        continue
          XOut=XOld+H
          if (XOut .gt. XF) then
            H=XF-XOld
            XOut=XF
          endif
          do JEqn=1,NEqn
            do iii=1,5
              YIterS(JEqn,iii)=YIterP(JEqn,iii)
              YIterN(JEqn,iii)=YIter(JEqn,iii)
            end do
          end do
c
c   Rescale iteration vector for new step size
```

```
c
        call rescale(YIterN,H,HOld,FStage,YIterS,NEqn)
        call Solver(XOld,NEqn,YIterN,H,HOld,F,ErrEst,
     1          FStageN,YIterS,Starting,YStage)
        NCalls=NCalls+5
        if (ErrEst .gt. Tol) then
          NTries=NTries+1
          if (NTries .gt. MaxTries) then
             print *,'At XOut=',XOut,', MaxTries exceeded'
             print *,'Tol=',Tol
            STOP
          endif
          NMisses=NMisses+1
          H=H/2
          goto 3
        endif
        HNew=NewStep(H,Tol,ErrEst)
        HOld=H
        H=HNew
        do JEqn=1,NEqn
          Y(JEqn)=YIterN(JEqn,1)
          do iii=1,5
            YIterP(JEqn,iii)=YIterS(JEqn,iii)
            YIter(JEqn,iii)=YIterN(JEqn,iii)
            FStage(JEqn,iii)=FStageN(JEqn,iii)
          end do
        end do
      if (.not. Diag) then
        goto 2
      endif
    endif
  end
```

```
SUBROUTINE Solver(X0,NEqn,YIter,H,HOld,F,ErrEst,FStage,YIterP,
   1            Starting,YStage)
c
c   Solver carries out one step of the solution for an explicit fifth
c   order DIMSIM ODE solver for an ODE system.  U is assumed to be the
c   identity matrix.
c
c***********************************************************************
c solver applies the basic DIMSIM algorithm to compute a new external stage
c vector and an error estimate.  Different coefficients are used in computing
c the error estimate if Starting is .true.
c
c     Parameters:
c X0:  (real*8) Starting point for this integration step
c NEqn,H,HOld,F,ErrEst,Starting:  See description of driver
c FStage:  (real*8) Upon output, set to vector of derivatives at internal
c       stage points for current step, dimensioned (NEqn,5)
c YIter:  (real*8) Upon input, set to rescaled previous external stage vector.
c Upon output, set to current external stage vector.  Dimensioned (NEqn,5)
c YIterP:  (real*8) Output only, set to input value of YIter.  Dimensioned
c       (NEqn,5)
c YStage:  (real*8) Temporary storage needed for each internal stage vector in
c       turn during calculation.  Dimensioned NEqn.
c***********************************************************************
*****
    implicit none
     real*8 X0,H,ErrEst,YIter(NEqn,5),HOld,Delta,Err
     real*8 YStage(NEqn),YIterP(NEqn,5),FStage(NEqn,5),Eps
      real*8 C(5),A(5,5),B(5,5),vT(5),W(5,6),BT(6,5)
    real*8 Beta(5),Gamma(5),vTYIter,Fac,BetInit(5),GamInit(5)
    integer iii,jjj,JEqn,NEqn
    logical Starting
    parameter (Eps=2.2d-16)
    external F
    common /Method/C,A,B,vT,W,BT,Beta,Gamma
    save Method
    Delta=H/HOld
     do JEqn=1,NEqn
       do iii=1,5
         YIterP(JEqn,iii)=YIter(JEqn,iii)
       end do
    end do
     call F(X0+C(1)*H,YIterP(1,1),FStage(1,1),NEqn)
    do iii=2,5
      do JEqn=1,NEqn
        YStage(JEqn)=YIterP(JEqn,iii)
        do jjj=1,iii-1
          YStage(JEqn)=YStage(JEqn)+
   1                 H*A(iii,jjj)*FStage(JEqn,jjj)
        end do
      end do
```

```
    call F(X0+C(iii)*H,YStage,FStage(1,iii),NEqn)
  end do
  do JEqn=1,NEqn
    vTYIter=0
    do iii=1,5
      vTYIter=vTYIter+vT(iii)*YIterP(JEqn,iii)
    end do
    do iii=1,5
      YIter(JEqn,iii)=vTYIter
      do jjj=1,5
        YIter(JEqn,iii)=YIter(JEqn,iii)+
1                 H*B(iii,jjj)*FStage(JEqn,jjj)
      end do
    end do
  end do
  ErrEst=0
  if (.not. Starting) then
    do JEqn=1,NEqn
      Err=0
      do iii=1,5
        Err=Err+H*Beta(iii)*FStage(JEqn,iii)+
1           Gamma(iii)*YIterP(JEqn,iii)
      end do
      if (abs(Err) .gt. ErrEst) then
        ErrEst=abs(Err)
      endif
    end do
    Fac=Delta**4/(18.82229382702224d0+1.079974531970991d2*Delta+
1      1.433089043343375d2*Delta**2+
2      5.513374496429978d1*Delta**3+Delta**4)
    ErrEst=Fac*ErrEst
  else
    BetInit(1)=0.1812875545254936d0
    BetInit(2)=0.3105254335256836d0
    BetInit(3)=-0.3258295105315199d0
    BetInit(4)=0.1138267699195702d0
    BetInit(5)=-0.01051030018603054d0
    GamInit(1)=-0.1351324309632461d0
    GamInit(2)=0
    GamInit(3)=0
    GamInit(4)=0
    GamInit(5)=0.1351324309632461d0
    do JEqn=1,NEqn
      Err=0
      do iii=1,5
        Err=Err+H*BetInit(iii)*FStage(JEqn,iii)+
1           GamInit(iii)*YIterP(JEqn,iii)
      end do
      if (abs(Err) .gt. ErrEst) then
        ErrEst=abs(Err)
      endif
```

```
  end do
endif
end
```

```
subroutine start(YIterP,F,X0,Y0,NEqn,H,YP,Tol,Y1,Y1P,Y6Der,
  1           Y2,Y2P,Y4,YNord,RKWork)
c*********************************************************************
***
c start first calculates the derivative at the initial point X0.  A scaling
c factor g to estimate the size of the initial Nordsieck vector is computed and
c this is used to determine h0.  In most cases g will be the smaller of the
c norm of the function and the first derivative, but in case one is smaller
c than 1000*eps (machine epsilon) the larger is taken, and in no case is g
c taken as less than 10-6.  A 6th order Runge-Kutta method is used by calling
c RK6 to compute the solution Y4 at X0+h0/4, Y2 at X0+h0/2 (from Y4), and Y1
c at X0+h0 (from Y2).  These are then used to calculate a Nordsieck vector for
c computation of the initial external stage vector, plus an extra derivative
c for computation of the appropriate size for the first DIMSIM step.  Both
c the Nordsieck vector and the intial external stage vector are returned in
c case tolerance is not met.
c YIterP:  (real*8) Initial external stage vector, output dimensioned (NEqn,5)
c F,X0,Y0,NEqn:  See dim51 description
c H: (real *8) Output value for calculated initial DIMSIM step size
c YP: (real*8) Derivative of the solution at X0, calculated using y'=F(X0,Y0),
c     output dimensioned NEqn
c Tol: (real*8) Initial tolerance computed as described in section a from
c     RTol, Y0 and ATol, input
c Y1,Y1P,Y2,Y2P,Y4:  (real*8 arrays) Values computed using 6th order
c     Runge-Kutta method for solution at h0, h0/2, and h0/4, and associated
c     derivatives computed using F, workspace each dimensioned NEqn.
c Y6Der:  (real*8 array) 6th derivative of solution vector at X0, multiplied
c     times  , workspace dimensioned NEqn.
c YNord:  (real*8) Nordsieck vector at X0 for step size H, output dimensioned
c     (NEqn,6)
c RKWork:  See driver description
c*********************************************************************
**
    implicit none
     real*8 YIterP(NEqn,5),X0,Y0(NEqn),H,YP(NEqn),Tol,YPP,norm,EPS
    real*8 Y1(NEqn),Y1P(NEqn),Y2(NEqn),g,h0,delta
     real*8 C(5),A(5,5),B(5,5),vT(5),W(5,6),BT(6,5),Beta(5),Gamma(5)
    real*8 Y6Der(NEqn),YNord(NEqn,6),Y2P(NEqn),Y4(NEqn)
    real*8 RKWork(8*NEqn)
    integer NEqn,JEqn,iii,jjj
    parameter (EPS=2.2d-16)
    external F
    common /method/C,A,B,vT,W,BT,Beta,Gamma
    save Method
    call f(X0,Y0,YP,NEqn)
c
c   Compute a value for h0
c
    g=dmin1(norm(Y0,NEqn),norm(YP,NEqn))
    if (g .lt. 1000*eps) g=dmax1(norm(y0,NEqn),norm(YP,NEqn))
    g=max(g,1.0d-6)
```

```
      h0=10**(5.0d0/6)*(eps/g)**(1.0d0/6)
c
c     Compute a value for y4 at X0+H0/4 using sixth order
c      Runge-Kutta method
c
       call RK6(Y4,X0,Y0,h0/4,F,NEqn,RKWork(1+7*NEqn),RKWork(1),
     1        RKWork(1+NEqn),
     1        RKWork(1+2*NEqn),RKWork(1+3*NEqn),RKWork(1+4*NEqn),
     2        RKWork(1+5*NEqn),RKWork(1+6*NEqn))
c
c     Compute a value for y2 and y2P at X0+H0/2 using sixth order
c      Runge-Kutta method
c
       call RK6(Y2,X0+h0/4,Y4,h0/4,F,NEqn,RKWork(1+7*NEqn),RKWork(1),
     1      RKWork(1+NEqn),
     1        RKWork(1+2*NEqn),RKWork(1+3*NEqn),RKWork(1+4*NEqn),
     2        RKWork(1+5*NEqn),RKWork(1+6*NEqn))
      call F(X0+h0/2,Y2,Y2P,NEqn)
c
c     Compute a value for y1 and y1P at X0+H0 using sixth order
c      Runge-Kutta method
c
       call RK6(Y1,X0+h0/2,Y2,h0/2,F,NEqn,RKWork(1+7*NEqn),RKWork(1),
     1        RKWork(1+NEqn),
     1        RKWork(1+2*NEqn),RKWork(1+3*NEqn),RKWork(1+4*NEqn),
     2        RKWork(1+5*NEqn),RKWork(1+6*NEqn))
      call F(X0+h0,Y1,Y1P,NEqn)
c
c     Compute sixth derivative times h0^6
c
      do JEqn=1,NEqn
        Y6Der(JEqn)=1280*(-90*Y0(JEqn)-22*Y1(JEqn)-144*Y2(JEqn)+
     1            256*Y4(JEqn)+H0*(-9*YP(JEqn)+
     2            3*Y1P(JEqn)+36*Y2P(JEqn)))
      end do
c
c     Compute initial step size
c
      H=h0*(Tol/(4*(5.518667640362375d-5)*norm(Y6Der,NEqn)))
     1            **(1.0d0/6)
      if (H .lt. 100*EPS) then
        print *,'START-Tolerance too tight or H too small'
           print *,'h0=',h0,'H=',H,'New H=1.0d-8'
        H=1.0d-8
      endif
c
c     Compute starting vector
c
      delta=H/h0
      do JEqn=1,NEqn
        YNord(JEqn,1)=Y0(JEqn)
```

```
      YNord(JEqn,2)=H*YP(JEqn)
      YNord(JEqn,3)=2*(-567*Y0(JEqn)-25*Y1(JEqn)-432*Y2(JEqn)+
     1         1024*Y4(JEqn)+H0*(-90*YP(JEqn)+3*Y1P(JEqn)+
     2         72*Y2P(JEqn)))/9
      YNord(JEqn,3)=YNord(JEqn,3)*delta**2
      YNord(JEqn,4)=1836*Y0(JEqn)+148*Y1(JEqn)+2112*Y2(JEqn)-
     1         4096*Y4(JEqn)+H0*(222*YP(JEqn)-
     2         18*Y1P(JEqn)-384*Y2P(JEqn))
      YNord(JEqn,4)=YNord(JEqn,4)*delta**3
      YNord(JEqn,5)=32*(-1323*Y0(JEqn)-169*Y1(JEqn)-1836*Y2(JEqn)+
     1         3328*Y4(JEqn)+H0*(-144*YP(JEqn)+
     2         21*Y1P(JEqn)+378*Y2P(JEqn)))/3
      YNord(JEqn,5)=YNord(JEqn,5)*delta**4
      YNord(JEqn,6)=160*(378*Y0(JEqn)+70*Y1(JEqn)+576*Y2(JEqn)-
     1         1024*Y4(JEqn)+H0*(39*YP(JEqn)-
     2         9*Y1P(JEqn)-132*Y2P(JEqn)))
      YNord(JEqn,6)=YNord(JEqn,6)*delta**5
     end do
c
c    We make use of the special form of W for c0=0, explicit methods
c
     do JEqn=1,NEqn
       YIterP(JEqn,1)=Y0(JEqn)
       do iii=2,5
         YIterP(JEqn,iii)=0
         do jjj=1,6
           YIterP(JEqn,iii)=YIterp(JEqn,iii)+
     1         W(iii,jjj)*YNord(JEqn,jjj)
         end do
       end do
     end do
     end
```

```
subroutine rescale(YIter,H,HOld,FStage,YIterP,NEqn)
c**********************************************************************
*
c rescale takes a previous external stage vector computed using a step size
c HOld and uses the DIMSIM rescaling algorithm to rescale it to become the
c appropriate previous external stage vector for continuing the computation
c with step size H.
c
c     Parameters:
c YIter:  (real*8) Output rescaled extermal stage vector, dimensioned (NEqn,5)
c H:  (real*8) Input new step length
c HOld:  (real*8) Input step length for previous successful step.
c FStage:  (real*8) Input vector of derivatives at stage points for last step
c          completed
c YIterP:  (real*8) Input previous external stage vector, calculated using step
c          length HOld, dimensioned (NEqn,5)
c NEqn:  See dim51
c**********************************************************************
*
   implicit none
   real*8 Yiter(NEqn,5),H,HOld,FStage(NEqn,5),YIterP(NEqn,5),Delta
    real*8 C(5),A(5,5),B(5,5),vT(5),W(5,6),BT(6,5),Sum1,Sum2
   real*8 Beta(5),Gamma(5)
   integer i,j,k,JEqn,NEqn
   common /Method/C,A,B,vT,W,BT,Beta,Gamma
   save Method
   Delta=H/HOld
   do JEqn=1,NEqn
     do i=1,5
       Sum1=0
       Sum2=0
       do j=1,5
         Sum2=Sum2+vT(j)*YIterP(JEqn,j)
       end do
       do j=1,5
         do k=1,6
           Sum1=Sum1+W(i,k)*Delta**(k-1)*BT(k,j)*FStage(JEqn,j)
         end do
       end do
       YIter(JEqn,i)=HOld*Sum1+Sum2
     end do
   end do
   end
```

```
real*8 function NewStep(H,Tol,ErrEst)
c*********************************************************************
**
c The error estimate is compared with the tolerance to produce a step change
c factor modifying the step size in preparation for the next step.  The local
c truncation error is assumed to be proportional to the 6th power of the step
c size.  The largest possible step change is a factor of 2.  A step change
c safety factor equal to .75 was used for DIMEX5 so that the step size used
c was .75 of optimal.  A larger factor of .9 was used successfully for DIMEX2.
c
c    Parameters:
c H:  (real*8) At input contains old step size.  At output contains new step
c          size
c Tol:  (real*8) Input tolerance derived from RTol, Y and ATol as described
c          in section a above
c ErrEst:  (real*8) Input current local error estimate
c*********************************************************************
**
   implicit none
    real*8 ErrEst,RTol,ATol,Tol,HFac,H,Eps,YNorm
   parameter (Eps=2.2d-16)
   HFac=.75d0*(Tol/abs(ErrEst))**(1.0d0/6)
    HFac=dmin1(HFac,2.0d0)
   NewStep=HFac*H
   end
```

```
subroutine NordInterp(Y,H,Theta,FStage,YIter,NordVec,NEqn)
c***********************************************************************
*
c The Nordsieck vector NordVec is calculated at the current value of XOut,
c using the previously calculated external stage vector YIter, the current
c step size H, and the current vector of derivatives at stage points, FStage.
c Theta then is used to rescale the Nordsieck vector to correspond to the
c desired output point and a 5th order Taylor series approximation Y is formed.
c
c    Parameters:
c Y:  (real*8 vector) Output approximation to the solution at the desired
c        output point, length NEqn.
c H:  (real*8) Input step size for last successful integration
c Theta:  (real*8) Input between 0 and 1, ratio (X-XOld)/H, where X is the
c        desired output  point
c FStage: (real*8 vector) Input vector of derivatives at stage points for
c        last successful integration, dimensioned (NEqn,5)
c YIter: (real*8 vector)  Input previous external stage vector, dimensioned
c        (NEqn,5)
c NordVec:  (real*8 vector) Workspace (output for waveform relaxation) for
c        Nordsieck vector at current XOut and last step size used.
c NEqn:  See dim51
c***********************************************************************
*
   implicit none
   real*8 H,Theta,FStage(NEqn,5),YIter(NEqn,5),Y(NEqn)
   real*8 NordVec(NEqn,6),Bt(6,5),vT(5),vTIter,del,C(5),A(5,5)
   real*8 B(5,5),W(5,6),Beta(5),Gamma(5),Factorial
   integer JEqn,NEqn,iii,jjj
   common /Method/C,A,B,vT,W,Bt,Beta,Gamma
   save Method
   do JEqn=1,NEqn
     vTIter=0
     do iii=1,5
       vTIter=vTIter+vT(iii)*YIter(JEqn,iii)
     end do
     do iii=1,6
       NordVec(JEqn,iii)=vTIter
       do jjj=1,5
         NordVec(JEqn,iii)=NordVec(JEqn,iii)+
   1                 H*Bt(iii,jjj)*FStage(JEqn,jjj)
       end do
     end do
     del=-(1-theta)
     Y(JEqn)=NordVec(JEqn,1)
     factorial=1
     do iii=2,6
       Y(JEqn)=Y(JEqn)+NordVec(JEqn,iii)*del**(iii-1)/Factorial
       Factorial=Factorial*iii
     end do
   end do
```

end

```
subroutine RK6(Y,X0,Y0,H,F,NEqn,YP,YP2,YP3,YP4,YP5,YP6,YP7,YTemp)
c**********************************************************************
***
c A simple 6th order Runge-Kutta solver with 7 stages, used to provide solution
c valuesneeded for starting method.  Solution Y for initial values (X0,Y0) is
c found at X0+H for problem Y'=F(X,Y).  The method is described in Butcher,
c pp. 203-205.
c
c     Parameters:
c Y:  (real*8 vector) Solution at X0+H, dimensioned NEqn.
c X0:  (real*8) Initial value of independent variable.
c Y0:  (real*8 vector) Initial value of dependent variable, dimensioned NEqn
c H:  (real*8) step size used
c F:  see dim51
c NEqn:  see dim51
c YP,YP2,YP3,YP4,YP5,YP6,YP7:  (real*8 vectors) Workspace used for storing
c              derivatives of stages, each dimensioned NEqn.
c YTemp:  (real*8 vector) Workspace used for storing stage vectors
c              successfully, dimensioned NEqn
c**********************************************************************
***
      implicit none
      integer JEqn,NEqn
      real*8 H,X0,Y0(NEqn),YP(NEqn),YP2(NEqn),YP3(NEqn),YP4(NEqn)
      real*8 Y(NEqn),YP5(NEqn),YP6(NEqn),YP7(NEqn),YTemp(NEqn)
      external F
      call F(X0,Y0,YP,NEqn)
      do JEqn=1,NEqn
        YTemp(JEqn)=Y0(JEqn)+H*YP(JEqn)/3
      end do
      call F(X0+H/3,YTemp,YP2,NEqn)
      do JEqn=1,NEqn
        YTemp(JEqn)=Y0(JEqn)+2*H*YP2(JEqn)/3
      end do
      call F(X0+2*H/3,YTemp,YP3,NEqn)
      do JEqn=1,NEqn
        YTemp(JEqn)=Y0(JEqn)+H*(YP(JEqn)+4*YP2(JEqn)-YP3(JEqn))/12
      end do
      call F(X0+H/3,YTemp,YP4,NEqn)
      do JEqn=1,NEqn
        YTemp(JEqn)=Y0(JEqn)+H*(25*YP(JEqn)-110*YP2(JEqn)+
     1            35*YP3(JEqn)+90*YP4(JEqn))/48
      end do
      call F(X0+5*H/6,YTemp,YP5,NEqn)
      do JEqn=1,NEqn
        YTemp(JEqn)=Y0(JEqn)+H*(18*YP(JEqn)-55*YP2(JEqn)-
     1            15*YP3(JEqn)+60*YP4(JEqn)+12*YP5(JEqn))/120
      end do
      call F(X0+h/6,YTemp,YP6,NEqn)
      do JEqn=1,NEqn
        YTemp(JEqn)=Y0(JEqn)+H*((-261d0/260)*YP(JEqn)+
```

```
1         (33d0/13)*YP2(JEqn)+(43d0/156)*YP3(JEqn)+
2         (-118d0/39)*YP4(JEqn)+(32d0/195)*YP5(JEqn)+
3         (80d0/39)*YP6(JEqn))
 end do
 call F(X0+h,YTemp,YP7,NEqn)
 do JEqn=1,NEqn
   Y(JEqn)=Y0(JEqn)+H*(13*YP(JEqn)+55*YP3(JEqn)+55*YP4(JEqn)+
1      32*YP5(JEqn)+32*YP6(JEqn)+13*YP7(JEqn))/200
 end do
 END
```

```
block data Matrices
c***********************************************************************
*****
c The matrices and vectors defining the particular method are set to
c appropriate values using data statements and stored in the common region
c /Method/.
c***********************************************************************
*****
      real*8 C(5),A(5,5),B(5,5),vT(5),W(5,6),BT(6,5),Beta(5),Gamma(5)
      common /Method/C,A,B,vT,W,BT,Beta,Gamma
      save Method
      data vT/-.2406956155386215d0,1.2604945758471451d0,
     1   -2.4812693924523267d0,1.9199070083032958d0,
     2   5.415634238405067d-1/
      data C /0,2.5d-1,5.0d-1,7.5d-1,1/
      data A/0,1.1765281703106688d0,1.9805793463233191d0,
     1   3.0532835108392395d0,1.4193325269467698d0,2*0,
     2   .4017181027378085d0,.7349961462028882d0,
     3   2.6534897473125331d0,3*0,.2672357626475791d0,
     4   -2.2778532945468265d0,4*0,1.1978905088172778d0,5*0/
      data B/3.16302391436455573 6d0,3.2501766921423335 14d0,
     1      3.5085280338489694 59d0,4.1762239549237720 36d0,
     2      3.0082207853047659 14d0,1.9743392460505406 810,
     3      1.5319781349394295 7d0,0.3273742041840276 25d0,
     4      -2.6182717193931199 2d0,2.1145334195877050 7d0,
     5      -0.8104251200556281 3d0,0.0979082132777052 03d0,
     6      2.2390605192329535 36d0,7.0390040987744303 7d0,
     7      0.3572048837825639d0,0.5409220188020184 01d0,
     8      -0.4222724256424260 43d0,-2.0974525093754521 55d0,
     8      -5.2884360132659356d0,
     9      -1.9042533085759860 4d0,0.0550786541963168 3844d0,
     1      -0.4613800716699075 171d0,-0.9368689835938225 39d0,
     2      -1.6910970273710501 62d0,-0.6456265552709995 2d0/
      data Bt/3.163023914364555d0,0,1,1.466666666666667d1,96,256,
     1      1.974339246050541d0,0,-5.333333333333333d0,
     2      -7.466666666666667d1,-448,-1024,-0.810425120055628d0,0,
     3       12,152,768,1536,0.540922018802018 4d0,0,-16,
     4       -1.386666666666667d2,-576,-1024,0.05507865419631683d0,
     5       1,8.333333333333333d0,4.666666666666667d1,160,256/
      data W/1,1,1,1,1,0,-0.9265281703106 69d0,-1.882297449061128d0,
     1       -3.305515419689707d0,-1.992859488529754d0,0,
     2       0.03125d0,0.02457047431554788d0,-0.0361169178745116d0,
     3       0.07713632883232162d0,0,0.002604166666666666d0,
     4       0.00827964262277682d0,0.01393940010021236d0,
     5       0.03157006827664394d0,0,0.0001627604166666666d0,
     6       0.001558025774120291d0,0.005702129564105415d0,
     7       -0.002014862315115681d0,0,8.138020833333333d-6,
     8       0.0001950328608825182d0,0.001161984318267553d0,
     9       -0.001959141967572072d0/
      data Beta /-2.385388840413769d3,-1.706352417381188d2,
     1         1.870907224366729d2,-2.323292678834744d1,0/
```

```
 data Gamma /1.268306923377163d3,0,-1.224572201881436d3,0,
1         -4.373472149572754d1/
 end
```

INITIAL DISTRIBUTION

1 Commander in Chief, U. S. Pacific Fleet, Pearl Harbor (Code 325)
1 Naval War College, Newport
1 Headquarters, 497 IG/INT, Falls Church (OUWG Chairman)
2 Defense Technical Information Center, Fort Belvoir
1 Center for Naval Analyses, Alexandria, VA (Technical Library)

ON-SITE DISTRIBUTION

4 Code 4BL000D (3 plus Archives copy)
1 Code 4B0000D
17 Code 4B4000D
    S. Chesnut (1)
    C. Schwartz (1)
    J. VanWieren (15)
1 Code 472000D
1 Code 473000D